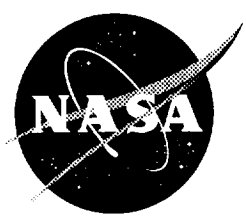
NASA SP-6101 (12)

ISSUES IN NASA PROGRAM AND PROJECT MANAGEMENT

Focus on Project Planning and Scheduling



NASA SP-6101 (12)

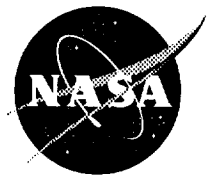
ISSUES IN NASA PROGRAM AND PROJECT MANAGEMENT

Focus on Project Planning and Scheduling

edited by

Edward J. Hoffman
Program Manager
NASA Program and Project Management Initiative

William M. Lawbaugh
Technical Writer and Editor
Technical and Administrative Services Corporation



National Aeronautics and Space Administration
Office of Management Systems and Facilities
Scientific and Technical Information Program
Washington, DC 1997

Issues in NASA Program and Project Management

National Aeronautics and Space Administration **Spring 1997**

Focus on Project Planning and Scheduling

Page

SP-6101(12) *Issues in NASA Program and Project Management* is twelfth in a series from NASA's Program and Project Management Initiative. This edition is collected and edited by Dr. Edward J. Hoffman with Dr. Pat Patterson and Dr. William M. Lawbaugh. Statements and opinions are those of the authors and do not represent official policy of NASA or the U.S. Government. Useful and enlightening material is welcome, and diversity of ideas is encouraged.

Inquiries should be directed to Dr. Edward J. Hoffman, Program Manager, Office of Training and Development, Code FT, NASA Headquarters, Washington, DC 20546-0001.

Planning and Scheduling Training For Working Project Teams at NASA

by F. G. Patterson Jr.

In 1988 the National Aeronautics and Space Administration began its Program/Project Management Initiative (PPMI), a curriculum of Agencywide training in systems engineering and systems engineering management. Since its inception, many courses have been offered. Sixteen courses are now offered on a regular basis, shown in Figure 1. Between 1988 and May 1996, PPMI conducted 294 courses and trained 6,368 people.

Each of the courses has been designed and prepared for an Agencywide audience and addresses specific issues that confront NASA management. One of the most basic project management skills is *planning and scheduling*. In even the most rudimentary performance, a manager must prepare an ordered list of tasks, allocate resources to each task, and prepare a schedule that is realistic enough to convince higher level management that proper controls are in place. Because of its importance, planning and scheduling is included as part of several PPMI courses. These courses present a methodology for planning and scheduling to a diverse NASA-wide audience of both civil servants and contract personnel.

Problems with Traditional Methods of Project Planning and Scheduling

Planning and scheduling is an activity that has much in common with the definition of product requirements, and although the similarities may be recognized, the activities are usually conducted much differently. In the generation of product requirements, the engineering community is increasingly alert to the need of working with a group of stakeholders that is thought to be representative of all active interests in the development of the product. Representing what he refers to as the viewpoint of the sociologist, M. Jackson (1995) describes the definition of a system as something that "has to be continually renegotiated subjectively between the various stakeholders, who all have their own agendas and perspectives." In most NASA projects, the efficiency of the requirements team approach is preferred to a canvassing approach. Thus, a requirements team of stakeholders is carefully picked, and a process of requirements engineering is carried out (Patterson, 1997). The result of the team approach is a specification that reflects the needs of all the members of the team.

APM	Advanced Project Management
CoF	Construction of Facilities Management
CBP	Construction of Facilities Best Practices
IPM	International Project Management
MPM	Multi-Project Management
PM	Project Management
PPS	Project Planning and Scheduling
PROGM	Program Management
REQ	System Requirements
SAM	Software Acquisition Management
SE	Systems Engineering
SPI	Software Process Improvement
TM	Task Management
TPM	Topics in Project Management
TSPM	Topics in Software Project Management
TTC	Technology Transfer & Commercialization

Figure 1. A current offering of PPMI courses.

Most planning and scheduling activities, on the other hand, are done by the project manager, who often has the "help" of a support contractor, sometimes referred to as a *planner*. The conscientious project managers who compose their own plan and schedule have the benefit of adjudicating every decision,

negotiating every tradeoff, and, indeed, of participating in every word and symbol in the documentation, thus taking ownership of the documentation and its contents. Now, while the dedication of such a project manager is commendable, this process limits the scope of the task to the best efforts of a single person.

There is no one right person or group who, to the exclusion of the others, can do an adequate job of planning and scheduling. We have seen again and again that the program or project manager cannot know, or even analyze the quantity and level of detailed data necessary to synthesize a comprehensive plan. Task managers, while collectively representing a broader scope that a single individual, do not speak for or understand the issues of other stakeholders, such as the user community. Scientists are primary customers at NASA, but they are focused on the problem rather than the solution. Engineers have the opposite bias and address the solution rather than the problem.

When plans and schedules are written by a single person or group, and in cases in which contractor planning and scheduling personnel are used, the community of stakeholders is sometimes asked to "review and approve" the work. However, such methods do not often get the investment, understanding, or adequate attention of stakeholders who may be overwhelmed by—or, indeed, may not even recognize their own inputs in—the technical and symbolic language that is commonly in use. Thus, in such cases, there can be little sense of *ownership* of the plans by the stakeholder community.

A more fundamental problem is that a systems engineering approach (Sage, 1992) to planning and scheduling requires attention to project variables in three dimensions (Figure 2):

1. Structure,
2. Function, and
3. Purpose.

While the best efforts of project management may bring structure and process to a project, without stakeholder involvement the purpose dimension is likely to be underrepresented. The result is inevitably reflected in faulty planning.

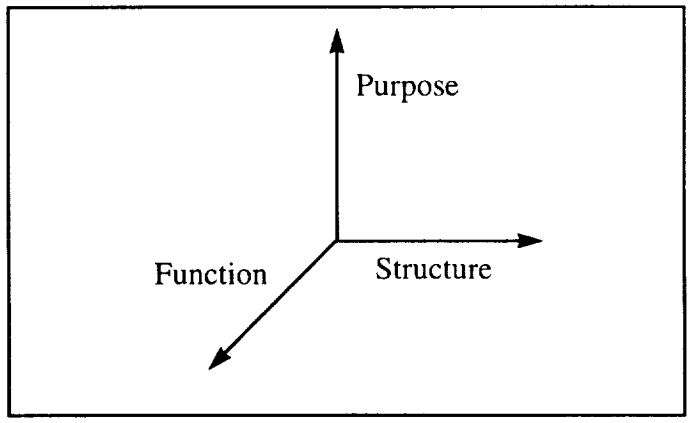


Figure 2. Three dimensions of planning and scheduling.

Problems with Traditional Methods of Project Planning and Scheduling Training

Traditional methods of planning and scheduling training use a "slide, lecture, demonstration, and exercise" format that does not engage the student adequately. In the best cases, fascinating case studies may be presented, in which important classes of problems are brilliantly analyzed and interpreted with the participation of the student. However, without the realism, and the attendant urgency offered by a project in progress, such exercises are little more than toys. In the worst case, students may be passive viewers of a "spectator sport." There is little investment and no urgency about the critical path or other results.

Moreover, the traditional "slide, lecture, demonstration, and exercise" format by its nature even in the best case fails to emphasize the most important aspects of planning:

- *Realistic negotiations among stakeholders* is unlikely. Planning, and replanning as a result of scheduling or other resource studies, is a process of "give and take" that loses effectiveness when it is merely "role playing" in a simulated negotiation in a traditional class setting.

- *The critical role of project manager cannot be realistically simulated*, except perhaps by a well prepared instructor who has thoughtfully studied the script (thus denying the students the opportunity to play the project manager role). There is no real basis for the project manager to decide among alternatives, since there is no reality to use for a reference.

- *Training may be unduly focused on automation*, since, of all the elements of the classroom exercise, the computer-driven process is the most realistic and most transferable to the participant's own project domain.

- *Inadequate training for identifying tasks and dependencies among tasks* is arguably the most elementary and important challenge of all.

NASA Project Planning and Scheduling (PPS) Training

Based in planning theory, NASA PPS training addresses fundamental needs that embody structure, function, and purpose:

- The need to allocate and structure resources (the *structure* dimension):
  - division of labor, positions;
  - structuring of time;
  - phasing of cost.

- The need to implement and to support an orderly process (the *function* dimension):
  - performance of tasks;
  - interrelationships among tasks;
  - roles of people and groups.

- The need to define, develop, and deploy a product that satisfies stakeholders in the project (the *purpose* dimension):
  - continual involvement of stakeholders;
  - availability of appropriate management controls;
  - attention to quality.

NASA PPS training focuses on the structuring of time and cost. As preliminary coursework (for which the Project Manager is responsible before the course meeting convenes), a work breakdown structure (WBS) is developed that will permit the identification of responsibility for the development of subsystems, including civil servants, contractor personnel, and their sub-contractors. Thus, the division of labor and the identification of positions in the project have been accomplished in advance, allowing the PPS training to address the division of time and cost.

During a PPS course, a team of stakeholders is assembled that includes the project manager and staff, subsystem managers and other task managers, customers (in NASA's case, these are often scientists), and experts in other areas whose contribution is essential to the success of the course. For example, an expert on project documentation is usually required. Depending upon the size of the project, the team size may vary greatly.

The basic task for the PPS participants is to determine and write down the tasks that need to be done, to create a partial ordering of the tasks that leads to successful completion of the project, to identify dependencies among tasks, to identify the person responsible for each task, and to estimate the resources required for each task. To accomplish the work of planning and scheduling, the representation of the tasks, their interrelationships, and their resource requirements is an important factor. We have two methods of representation that are currently in use for PPS training, depending upon the size of the project. For smaller projects, we use a Cards-on-the-Wall format that creates a network of resource-loaded tasks using cards to represent tasks and colored string between cards to represent dependencies. Each stakeholder sub-team has its own color for cards. This "life size" representation and color coding of the network allows stakeholders to navigate the walls, inspecting paths of special importance, bringing events of the future into the present where they may be purposefully influenced. For larger projects, we use the "one-pager" (Schoenfelder, 1995) representation.

Method for Smaller Projects

Our PPS course was developed by the Center for Systems Management in Cupertino, California. The course follows the following basic steps:

1. Identification of stakeholders.
2. Commitment to 4-day, 96-hour, off-site meeting with a single goal.
3. Using a WBS, identifications and ordering of project tasks by functional teams.
4. Identification of dependencies among project tasks.
5. Cards-on-the-Wall technique for displaying ordered tasks and dependencies.
6. Approval of network by project manager.
7. Capture of network into automated project management system.
8. Computation and analysis of critical path.
9. Tradeoffs of resources and goals.
10. Repetition of process to create a successful plan and schedule.

PPS training is different from a simple "facilitated meeting" in which a facilitator captures ideas and tries to assist in forming consensus among group members. PPS training uses a format in which the project manager *presides* over the process, but in which the leader *conducts* the process. It has proven to be essential to keep these roles distinct. That is, the project manager must not conduct, and the PPS process leader must not preside. The project manager is responsible for the correctness of the planning, for all assignments of responsibility, and for all other decisions about the *project*. The leader, on the other hand, is an expert on the PPS process and brings efficiency, objectivity, and closure to the meeting, but may know very little about the technical domain of the project being planned. The choice of a leader who can conduct and control the meeting is essential to its success. At NASA this separation of roles has been used very effectively.

Method for Larger Projects

To date, NASA PPMI has had only one experience with a large group of more than 200 people. Our approach used the "one pager" representation format, as previously mentioned. While the "cards on the wall" process undoubtedly scales up for use in larger groups, project managers may wish to use other representation formats for capturing information. For large projects, a recursive system of systems approach is used, in which parallel project planning and scheduling efforts are carried out for the smaller systems.

Beneficial Side-effects of PPS Training

Based upon surveys, participation, and personal observation, there is no doubt that each of the student participants in a PPS training session leaves with a new definition of *planning and scheduling*; a deep appreciation of the basic tools, including GANTT charts, PERT charts, logic networks, critical path analysis, project resource estimation, and automated tools; a personal success story that serves as a model for future planning activities; and an appreciation of the need for and the benefits of good planning. From the viewpoint of the NASA Office of Training and Development, these factors alone justify the use of the intact team approach as a training vehicle.

Moreover, at least four predictable side-effects are extremely beneficial to projects and have made PPS training very popular among knowledgeable project managers. They are:

1. **Team building.** Without exception, every PPS class has reported strongly effective team-building activity, recognition of the needs of other stakeholders, and improved understanding of and appreciation for product requirements.
2. **Identification of high-risk project plan elements.** Teams are compelled to recognize neglected or hard-to-face areas (often software),

understand interactions among tasks, and perceive relationships to critical paths. For example, in one project in which software had been largely ignored, the entire software documentation list was defined, planned, and scheduled during the training, an activity that resulted in identifying software development as the critical path.

3. **Reprogramming of inefficiently used resources.** Each of the most critical resources of a project—time, money, and people—is the object of careful scrutiny by a group of stakeholders, whose interests in the project (the purpose dimension) are at the forefront of their attention.

4. **Recognition and resolution of potential future problems.** By structuring a project plan that extends well into the future to the point of project completion, many errors and omissions may be corrected in the present, eliminating a future cost impact to the project.

Project Planning and Scheduling training with intact teams has been beneficial for students, projects, and the Agency. The intact team format has been used very successfully for more than a dozen smaller projects (25 or fewer participants) over a period of 18 months. It has also been used extremely successfully for one larger project (more than 200 participants).

Because of the PPMI's success with PPS training techniques, training with intact teams is being investigated for use in other program and project management needs. In particular, there are two candidate training programs whose team orientation suggests the intact team approach. They are requirements definition and software process self-assessment.

References

Jackson, M. (1995). *Software Requirements and Specifications*. Wokingham, England: Addison-Wesley Publishing Company, Inc.

Patterson, F. G., Jr. (1997). Chapter 1—Systems Engineering Life Cycles: Life Cycles for Research, Development, Test, and Evaluation (RDT&E); Acquisition; and Planning and Marketing. In A. P. Sage & W. Rouse (Eds.). *Handbook of Systems Engineering and Management*. New York: John Wiley & Sons, Inc.

Sage, A. P. (1992). *Systems Engineering*. New York: John Wiley & Sons, Inc.

Schoenfelder, T. E. (1995, Spring). The "One-Pager" Methodology and Application. *Issues in NASA Program and Project Management* (SP 6101(09), 1-14.

Warfield, J. N., & Hill, J. D. (1972). *A Unified Systems Engineering Concept*. Columbus, Ohio: Battelle Memorial Institute.

Project Planning and Scheduling Workshops: An Overview

by W. M. Lawbaugh

A highly acclaimed and well-received new training effort on the part of NASA's Program/Project Management Initiative (PPMI) has been taking shape over the past couple of years.

So far, about a dozen Project Planning and Scheduling (PPS) workshops have been completed. Each has been designed to provide project teams with an understanding of the principles of planning and scheduling, along with an opportunity to apply those principles to their own current project.

NASA staff and their contractors are brought together for four or five days (and late nights) to work on a project in the early planning or replanning stages. Project teams execute the fundamentals of planning, create and use a methodical work breakdown structure (WBS), and develop some kind of project logic network. From there, they generate a project schedule, and usually the definition and management of the critical path. Throughout the workshop the project team is expected to apply the principles of effective planning and scheduling in a hands-on effort for their current project.

Project Planning and Scheduling workshops are conducted on an as-needed basis at various sites for intact project teams, including NASA staff, customers and contractors. In order to develop a high-level integrated network with a calculated critical path, participants are asked to prepare for the planning process on two levels.

The first, essential level of preparation calls for a team leader, usually the NASA project manager, to work with a PPS facilitator and knowledgeable people who are responsible for the project. Upon arrival at the training site, the project team should have a detailed description of project objectives and control, along with a list of project milestones and deliverables, both internal and external.

First-level preparation also calls for computer hardware and software such as Microsoft Project to capture the project team's critical path at the end of the PPS workshop. An expert operator, furnished by the project team, is expected to handle up to 400 tasks, process all the data generated by the team, meet the online needs of the group, and then print out the project network.

A second level of preparation is advised to assure success of the workshop process. It is a good idea, for example, to create a pictorial illustration of all the essential components and interfaces of the project. A flow chart should show how those components are related to other systems. A hierarchical diagram should show the decomposition and integration structure, while an organizational diagram could illustrate the reporting structure of the project team. A list of constraints on the project would be helpful, along with a description of any strategy for project delivery.

To make sure the project managers, engineers and technicians are all speaking the same language, both a project glossary and list of acronyms are suggested. Often these lists are supplemented during the Project Planning and Scheduling workshop as it progresses.

Space Station Support Equipment (SE) Planning, Scheduling and Integration

One of the first PPMI Project Planning and Scheduling workshops involved the Space Station Support Equipment Integrated Product Team (IPT) from the Kennedy Space Center. Larry Manfredi

served as project manager and leader of the PPS workshop. The KSC support equipment is developed for the processing of International Space Station flight hardware resupply and return missions. The KSC support equipment IPT faces daunting challenges in terms of planning, scheduling and integration. The team will design, procure, and conduct verification of more than 75 end items of support equipment. Their task also includes the continuous coordination of interface control documents, design/document reviews, schedules and deliverables pertaining to more than 49 end items of non-KSC-developed support equipment to be turned over to the IPT for sustaining engineering.

The purpose of the PPS workshop was to ensure that members of the Communication & Avionics Sub-IPT, Simulators Sub-IPT, Electrical & Instrumentation Sub-IPT, the Test, Control and Monitor System (TCMS) IPT, and Logistics and Maintenance IPT would integrate their planning and scheduling for the U.S. International Standard Payload Rack (ISPR) Checkout Unit development. The ICU provides a sufficient fidelity test station, which will be used to verify that the ISPRs and EXPRESS (Expedite the Processing of Experiments to Space Station) racks are electrically and mechanically compatible with the space station module prior to prelaunch installation. The Integrated Product Team approach is used to ensure that empowered teams, staffed and supported by functional organizations, are accountable for designs that fully meet customer requirements and expectations. The team is responsible for requirements definition, design development, acquisition, fabrication, verification, training, operations support, maintenance, configuration accounting, and sustaining engineering of standalone end items and systems that must be integrated in order to complete the ICU.

The team members were given instructions on the Support Equipment IPT's technique of using concur-

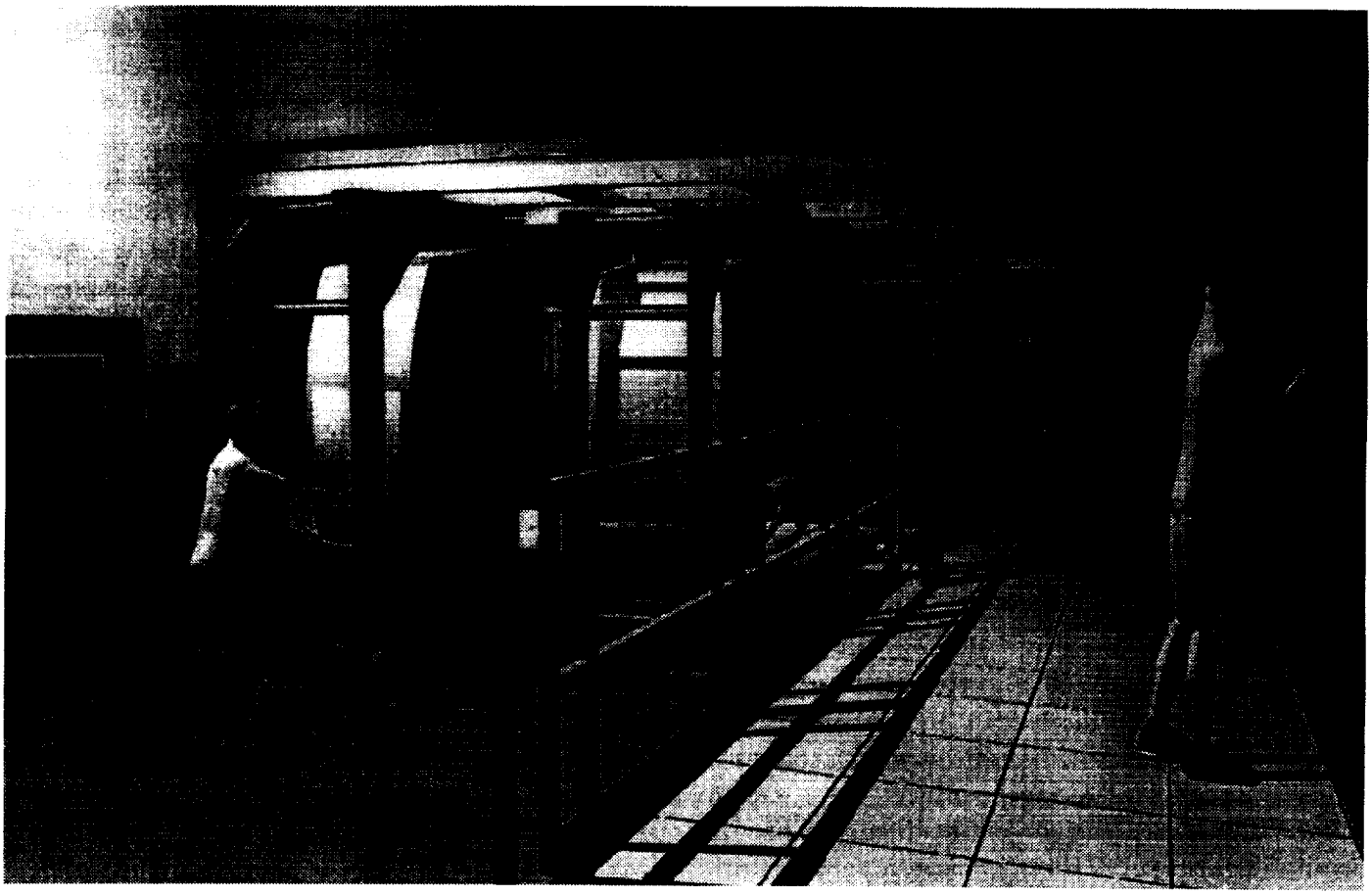


Figure 1. Space Station Support Equipment checkout unit.

rent engineering and integrated process-based management flows to facilitate planning and implementation. The End Item teams were briefed on the structure of the development process, which facilitates continuous improvement by incorporating all required products, activities and associated constraints into an automated project management/ scheduling tool. Each product being developed by the team was identified at the task level, along with required duration, input/output requirements, and interdependencies. Required skills were identified and assigned at the task level. Constraints were identified to facilitate Critical Path Method analyses. The planning and actual cycle time of each activity and product development will be traced to facilitate validation of future planning and Root-Cause Analysis.

As the team began to link interdependencies externally and internally, it became evident that there was a need for a more structured activation/validation plan to verify all interfaces in the ICU, including services from the Communication & Tracking Checkout System, Command & Data Handling, Power, Fluids and TCMS. This structured plan evolved as an integrated test scenario known as the Payload Integration Checkout Facility. The PICF is designed to integrate experiments and carriers such as ISPRs and perform a final interface verification test utilizing the TCMS and all other supporting subsystems.

All in all, the multi-disciplined composition of the End Item teams, along with the many international customers that utilize the ICU to accomplish their payload and experiment processing needs, says Michael Jones, makes the KSC Support Equipment Integrated Product Team's implementation task a unique challenge for effective project planning, scheduling and integration.

Stratospheric Aerosol and Gas Experiment III (SAGE III)

SAGE III comes from a long lineage of successful Langley Research Center SAGE-series programs. Three of the four previous instruments operated beyond their design-life and none has failed in-orbit. The fourth, actually the first instrument in the series, was operated for only four orbits during the Apollo-Soyuz mission in 1975 to establish measurement validity of the newly invented solar occultation concept. Two of the four instruments were operated beyond 14 years, with SAGE II still operating today and returning good science measurements. Each successive instrument added new spectral channels, but older instruments were kept operating to preserve the long-term data set. The SAGE series has the longest term data set for aerosols and ozone in the middle atmosphere, and is considered by the World Meteorological Organization to be the standard for global ozone and aerosol profile measurements.

SAGE III, like its predecessors, will be a principal source of data for global changes in aerosols, ozone, water vapor and clouds. State-of-the-art Charge Coupled Device (CCD) detector technology has been employed to boost sensitivity and spectral resolution. Increased sensitivity allows solar occultation measurements to be taken deeper in the troposhere to determine long-term global warming or episodic climate cooling after volcanic eruptions on Earth such as the 1991 Mount Pinatubo disturbance, and additionally, allows for lunar occultation measurements. Using lunar occultation, SAGE III measures nighttime species such as chlorine dioxide.

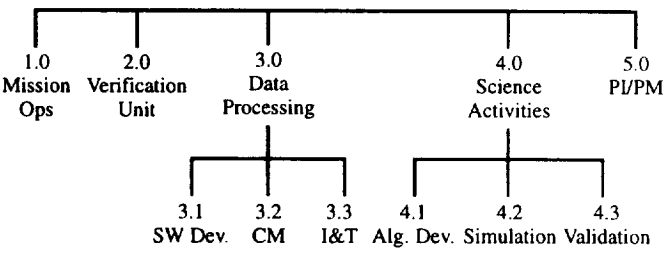


Figure 2. SAGE III WBS.

SAGE III is currently planned for multiple launches as part of the Earth Observing System. The first instrument will fly on a Russian spacecraft—METEOR 3M—in 1998. NASA Headquarters is currently negotiating with space agencies of other countries to find a home for the second instrument. An International Space Station mission beginning in 2001 is planned for the third instrument. International aspects of this program place special challenges on the SAGE III Team. Each team mem-

ber must be open not only to different cultures and new technical concepts, but to new ways of doing business that are very different from the American norm. Virtually every aspect of the Russian interface (personal, technical and programmatic) is vastly different from past experience. These challenges have the greatest effect on team efficiency. Thus, project work planning must include huge inefficiency factors to account for cultural differences, such as the language barrier where all discussions with the Russians must go through interpreters.

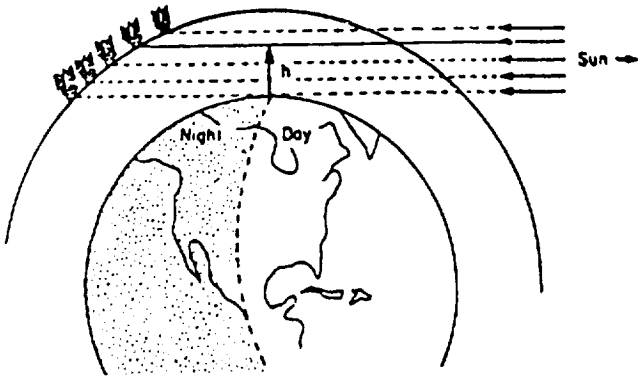


Figure 3. SAGE III measurements.

SAGE III was very fortunate in being able to schedule a PPMI Project Planning and Scheduling Workshop in Hagerstown, Maryland to coincide with the first week of the hardware development (Phase C/D) program. Twenty-seven team members representing Langley, Goddard, Wallops, and Headquarters civil service, on-site Langley contractors, and the prime contractor, Ball Aerospace, met during the second week of January 1995. Not just engineering team personnel, but everyone associated with the Project was invited to attend. During the first evening, sub-teams were organized to divide planning of overall team activities into smaller groups categorized by instrument subsystems, interfaces, operations, etc. Each sub-team planned its piece of the program for two days, and then reconvened as a team to integrate activities on the last two days. One of the most popular of the team building exercises was a meeting that lasted several hours early in the week, in which each statement, and each requirement in the government contract with Ball Aerospace was challenged. Each requirement and each deliverable to the government, including documents, had to meet a strict test: if it didn't contribute to measurement of ozone and aerosols in the atmosphere, it was thrown out. Needless to say, many statements and requirements were eliminated.

CSM facilitator John Chiorini helped the team organize the work into a detailed work breakdown structure (WBS), and indicated the time-phased, interrelated activities using yarn and the Cards-on-the-Wall approach. As each captain described the sub-team's plan, critiques from members of the other sub-teams served to brainstorm activities and interrelationships until yarn stretched completely around the large room and into smaller rooms at the back to describe relationships among the approximately 400 activities. The first critical path to be calculated indicated that delivery of the flight instrument was 14 months after the 34-month requirement. Subsequently, the team brainstormed more efficient logic to establish a plan to deliver flight hardware on time.

It was not surprising that the newly formed team began the week as an amorphous group of strangers with only a vague understanding of what SAGE was all about, but ended the week functioning as a high-performance team with a good work plan. According to Ed Mauldin, SAGE III Project Manager and Hagerstown team leader, the most important benefit from the week was quick development of new interpersonal relationships among team counterparts and establishment of a high-performance team very early in the program. Being off-site in an informal environment made it easy to forget who was government and who was contractor, thus eliminating useless communication barriers. A united team dedicated to building the best possible scientific instrument within budget and schedule constraints was formed and a common sense of purpose was instilled. Now, about halfway through the program, this team remains within budget and on schedule, a remarkable success story. This team is very proud of its record of establishing new standards for others to follow and highly recommends this PPMI Project Planning and Scheduling workshop process to other newly formed project teams.

Transport Research Flight Facilities

The third PPS workshop involved a diverse team of engineers, designers, computer hardware and software experts, QA, fabrication and resource analysts, schedulers and project management people headed by Allen C. Royal of Langley Research Center.

Their task was to plan and schedule the modification of a B-757 aircraft from an airline configuration to a research facility. In addition, the project team was expected to develop an instrumentation integration laboratory and create a simulator facility to replicate the aircraft research flight deck.

"The team needed the time away from the everyday working environment," said Royal, "to concentrate exclusively on the job at hand, which was to develop logic diagrams, work breakdown structures, GANTT charts, resource assignments, etc."

He added: "In addition, the time spent 'locked up' in a room 12 to 14 hours a day actually resulted in a closer knit group of people (very important, considering the job at hand)."

The four-and-a-half day experience brought the Langley team closer together with specialists from Lockheed, PSI, Unisys and CSC, Computer Sciences Corporation. "One of the many positive results of this experience was that as the individual teams worked," noted Royal, "people began to realize just what was expected of them and what they were to expect from another team, and the enormity of the overall project—this was a big plus."

Another big plus was the momentum that was built up during the PPS workshop that propelled the project past its first major internal milestone. This project team, too, asked for another PPS workshop but the principal players could not be scheduled at the same time.

Guidance, Navigation and Control Integration and Test Facility

The next PPS workshop was designed for the guidance, navigation, and control (GN&C) group developing a test facility for the International Space Station (ISS) of Johnson Space Center. The ISS GN&C function is distributed not only among different segments of the ISS, but between U.S. and Russian hardware and software. The GN&C Integration and Test Facility (GITF) was proposed by JSC Engineering as a facility where a majority of these pieces could be integrated and tested during development to increase the likelihood of the success of the on-orbit configuration.

GITF is bringing together all of the U.S. GN&C components to perform real-time closed loop testing. Flight-equivalent processors for both the GN&C and the Command & Control software will be integrated with the Global Positioning System (GPS) receiver processor, being fed inputs from the GPS radio frequency signal generator, the engineering unit rate gyro assembly, mounted on a three-axis rate table; and an emulator, which is being developed and built at JSC, of the Control Moment Gyro.

The Russian portion of the GN&C system will hopefully be represented by development units of the flight processors, being provided to the Russians by the European Space Agency, loaded with both development and final versions of the Russian flight software, and high fidelity models of the Russian sensors and effectors.

Project manager and group leader Karen Frank of JSC faces the challenges of relying on international cooperation for significant deliverables to her project, as well as the integration of institutionally owned resources with program-contracted hardware. Since the original workshop was conducted, numerous deliveries to the project have slipped schedule and the team has conducted its own mini-workshop, based on the PPS experience, to re-network and replan the project.

The next two Project Planning and Scheduling workshops occurred simultaneously but by different facilitators in September 1995. Blackhawk Management Corporation led the High-Speed Research planning and integration workshop in Hampton, Virginia, and CSM, the Center for Systems Management of Cupertino, California, facilitated the AGATE work-

shop in Hagerstown, Maryland. The two different approaches are detailed here.

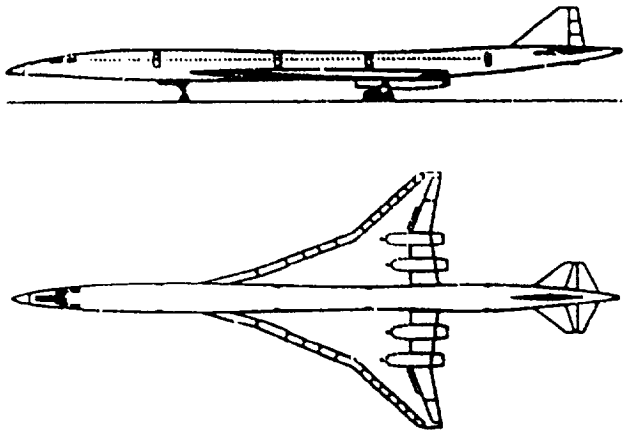


Figure 4. HSCT prototype.

High-Speed Research Program

For more than a quarter of a century, NASA has sponsored research for a supersonic transport aircraft. Environment concerns in the early 1970s led to a halt in funding while the British-French Concorde program moved forward.

A decade ago NASA received funding for Boeing and McDonnell Douglas to conduct studies of a second generation SST to carry about 300 passengers and flying 6,000 nm.

Phase 2 of the NASA/industry effort to develop the technology for the nation's first high-speed civil transport (HSCT) shifted into high gear with the High-Speed Research (HSR) planning and integration workshop held at the Chamberlain Hotel on Fort Monroe in Hampton, Virginia, in September 1995. More than 168 participants were present at the workshop, including officials and engineers from three NASA Centers (Langley, Ames and Lewis) and Boeing, Douglas, Lockheed and Northrop. The workshop, supported by NASA Headquarters under the Program/Project Management Initiative, used NASA expertise and the Blackhawk Management Corporation to teach the HSR Integrated Technology Development teams the latest advances in project management and planning skills. Rob Calloway of Langley was the NASA group leader. Specific tools presented to the attendees included the "One-Pager" tracking methods, logic networks and team building approaches.

In early 1995, Joe Shaw, Project Manager for the Propulsion segment of HSR at LeRC, and Dan Walker, Business Manager, sponsored a different approach to the application of the One-Pager concept to the HSR project. Rather than utilizing the workshop format, Joe Shaw formed a small team comprising, among others, James Wilcox of Blackhawk Management Corporation and Lisa Vietch of LeRC, to analyze the available data and develop the One-Pager products. This was successfully accomplished, and early returns suggest that the concept has proved to be very useful. (The One-Pager illustrations in this article are from the Propulsion segment of the HSR project at LeRC.)

At the PPS workshop, the high-speed research agenda for the next three years was set regarding HSCT airframe development. The workshop involved the efforts of 16 NASA/industry teams representing the following areas of study: structures and materials, aerodynamic performance, flight deck technology, environmental impact and overall technology integration. Phase 1 of the HSCT development program, involving technical solutions for environmental concerns, were completed later that year. Phase 2 of the program was fully implemented that year and addresses the cost effectiveness and economic viability of the aircraft systems.

The HSR program was spending approximately $20 million a month on HSCT research. NASA facilities, including advanced computer simulators, wind tunnels and labs, were being utilized to develop an HSCT technological database. As stated by Dr. Alan Wilhite, Deputy Director of the High-Speed Research Project Office at NASA Langley, "Technology is being developed for industry use in the year 2001."

The One-Pager approach involves a concise, integrated, executive level set of cost, logic, schedule and metrics data that encourages communication of plans and of progress against plans. This approach focuses on definitive end products with one or more

of these characteristics: high cost, high schedule risk, high technical risk and/or key integration intersection. (Weeding out less important items is extremely difficult, say the facilitators.) It starts with an understanding of intermediate level logic flow: "If you can't represent your area in one readable chart, you have too much detail." The approach relies not on milestone density but rather on defining schedule activities that can be communicated.

Implementation of the One-Pager concept calls for the imposition of certain intermediate level requirements on the technology manager in order to satisfy the requirement of consistency. While it requires a defined interface with detailed cost, logic, schedule and metric plans, it does not impose specific requirements on how a director manages below defined interfaces, such as a formal performance measurement system or low-level logic. Automation is desirable but not mandatory—communication is the key, and no known software can yet meet the conciseness and integration requirements.

Earned value computation with the One-Pager is somewhat subjective. Earned value is estimated at a high level and does not depend upon milestone counts. The plan is rebaselined only once a year unless otherwise directed, and earned value is computed against the baseline, not updated for changes. Thus, there are no "who's at fault" implications in the One-Pager approach.

The One-Pager concept is a proven methodology which should be given serious consideration for use in both very large hardware development projects and technology projects. It was developed by Phil Shanahan and James Wilcox in Texas and refined by NASA.

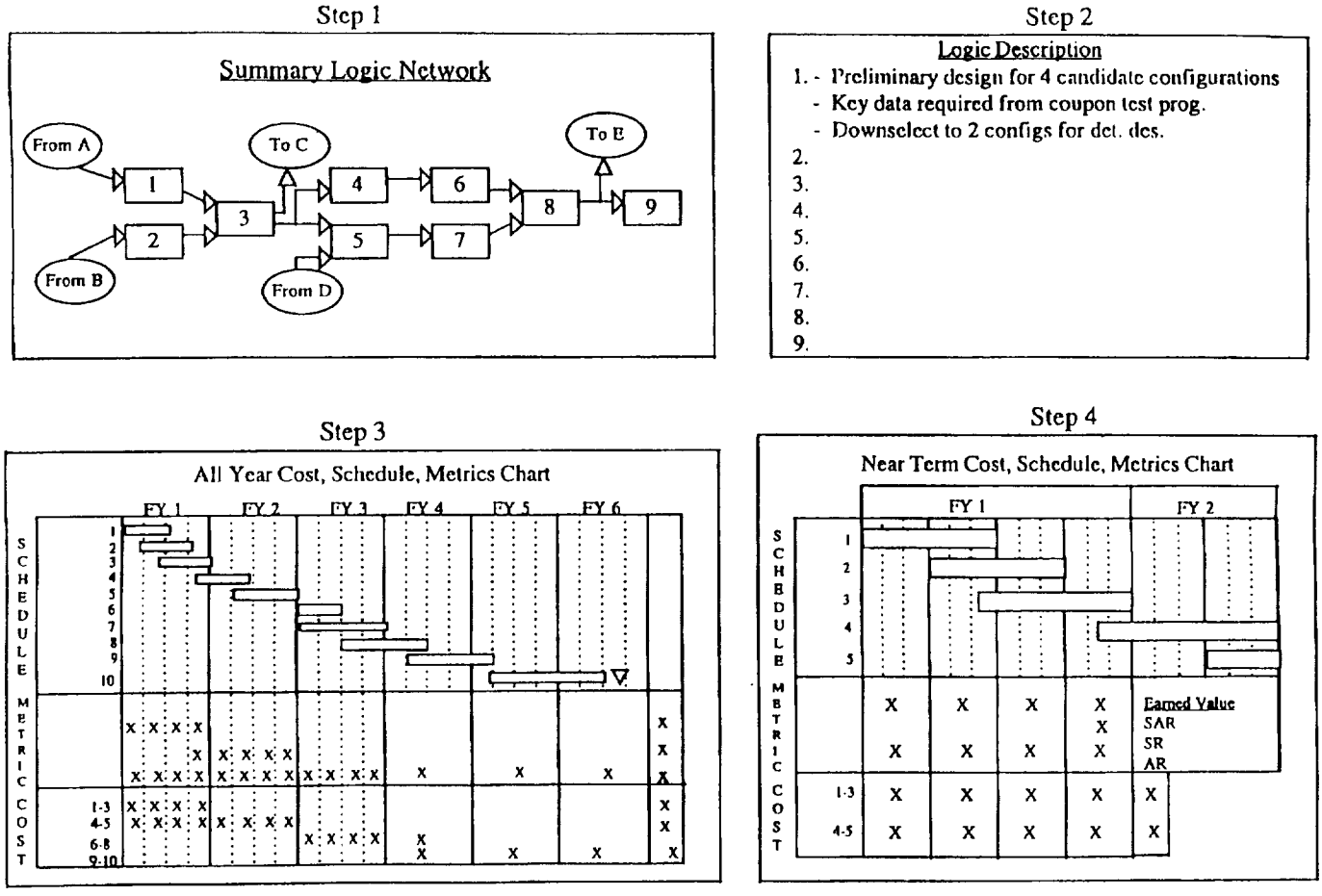


Figure 5. The One-Pager approach.

The AGATE Project Cycle

The Advanced General Aviation Transport Experiments (AGATE) project team in Hagerstown, Maryland, took a different approach with CSM facilitators. A stakeholder team approach to project planning and scheduling involves a Cards-on-the-Wall approach pioneered by Kevin Forsberg and Hal Mooz in California.

The purpose of the CSM workshop is to create a high-level integrated network with a calculated path.

The first effort is to develop a coherent Work Breakdown Structure (WBS) upon the foundation of a Project Products List. The PPL is a complete list of hardware, software, support equipment, support services, tools and documentation required to perform the contract. The WBS is broken down into manageable work packages that can be scheduled, budgeted, organized, statused and controlled.

Networking and scheduling are then introduced for a Project Master Schedule reflecting any requirements fixed by the customer. The Project Master Schedule usually includes project completion dates and customer-imposed reviews such as preliminary and critical design reviews, document delivery dates and the like.

The Critical Path Analysis is at the heart of the "Cards on the Wall" approach. A "Task Planning Form" is filled out and tacked or taped on the wall. Colored strings or yarn run from card to card showing "input" and "output" (expressed in nouns), connected to a "Task Description" expressed in verbs. Thus, "data" might connect to a verb such as "draft" with a noun output such as "report." The strings rep-

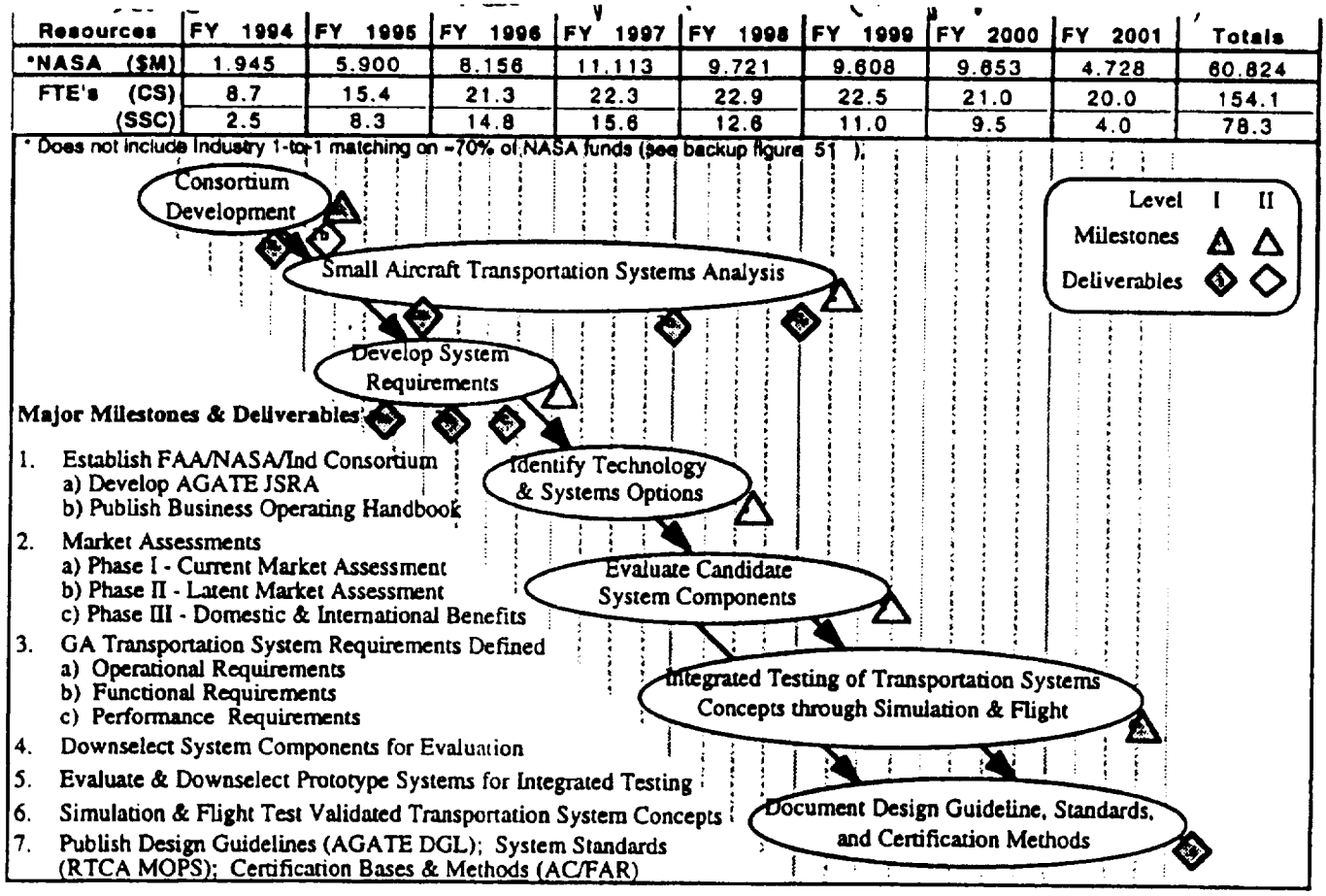


Figure 6. AGATE's baloney chart.

resent various tasks that feed into and flow out of major milestones and deliverables along a timeline.

General Aviation manager Bruce J. Holmes of Langley Research Center led the project team from Langley, Lewis, Avrotec of Oregon, Kestral of Oklahoma, Lockheed Martin, the National Institute for Aviation Research and Raytheon of Kansas, the Research Triangle Institute, Rockwell and Hamilton Standard.

After lectures on WBS development, networking and scheduling, and critical path analysis, the project team of 25 established assumptions and ground rules. Holmes presented the AGATE program roadmap showing the formation of a consortium among NASA, the FAA and the small aircraft industry. Following market analyses and general aviation system requirements, the AGATE group hopes to identify technology options, evaluate options, evaluate candidate system components and publish a library of documents for a revitalized small aircraft transportation system in America by the year 2001.

The AGATE project will require government and industry coordination in five work packages: flight systems, propulsion sensors and controls, integrated design and manufacturing, icing protection systems, and a new one, the AGATE integration platforms. Most of the facilities, such as simulators and laboratories/computers, are furnished by Langley. Lewis is furnishing the icing tunnel, and industry/university facilities are scheduled for flight tests.

SAGE III Science Plan

A year after the SAGE III project team met in Hagerstown for Project Planning and Scheduling, the project's science team met to coordinate the efforts among four contractor groups and two NASA Centers. Science Manager Lelia B. Vann of Langley Research Center led the project team from Langley, Goddard Space Flight Center (and Wallops Flight Facility), CSC, GATS, SAIC and IDEA, Inc.

The SAGE III is scheduled for launch in August 1998 on a Russian Meteor 3M spacecraft as part of NASA's Mission to Planet Earth (MTPE) program.

The SAGE III science team included algorithm development, software development for data processing, simulations, validation and mission operations. The team began with a detailed Work Breakdown Structure and ended up with a critical path. Some questions asked included: "What work needs to be done? Who will do it? How long will it take? What will it interface with?" Each task was assigned an estimate of labor, material and other resources. By focusing on critical path tasks, the project team can identify those sequences that will most likely determine the duration and drive the schedule of the project.

The LaRC SAGE III Principal Investigator (P.I.) is responsible for the science research activities, algorithm development, data processing, validation and mission operations. The MTPE program office is responsible for overall coordination of the mission, including funding, program integration and reporting on investigation. They will support SAGE III's communications, ground receiving station, and data generation and distribution.

To show the critical path for this multi-year project, CSM facilitator John Chiorini generated a chart at least 12-feet long showing the relationships of tasks among different organizations. So, why plan? His response: "To bring the future into the present so you can do something about it."

There is every indication that the SAGE III teams, as well as the other Project Planning and Scheduling workshop teams, will not execute their efforts exactly as conceived. Funding irregularities, management structure changes, personnel shifts and unforeseen events will inevitably alter their One-Pager and critical paths. That is to be expected.

What each of these project teams have, however, is a sense of direction. Team members know up front what the project will cost in terms of payroll, facilities and equipment. Any subsequent trade-off in any of the estimated resource areas will, they know, cost the project in terms of budget, schedule or performance. It may even derail the project if the trade-off is excessive.

Another thing each of these project teams now shares is camaraderie, if not just a better understanding of each other and the needs of each component in the project. For some projects, the PPMI Project Planning and Scheduling workshop was the first time all the major players came together in one room at the same time. That intangible, in and of itself, is invaluable, especially in an era where teamwork is the single most cited component of success in completed missions.

ID	Task Name	Duration	Start	Finish	Resource Names
1	**1.1 Mission Operations**	**510d**	**03/11/96**	**03/20/98**	
2	**1.1.1 DAAC /NOC Interface**	**60d**	**03/11/96**	**06/03/96**	
3	1.1.1.1 Data Transfer for Protocol	2w	03/11/96	03/22/96	PD[0.2],SN[0.2]
4	1.1.1.2 Support ECS Level 0 ICD	2w	05/20/96	06/03/96	MC[0.5],SN[0.5]
5	1.1.1.3 Determine Level 0 Took Kit Specification	2w	03/25/96	04/05/96	SN
6	1.1.1.6 Determine Definitive Orbit Format and Meta Data	1w	05/13/96	05/17/96	SN
7	**1.1.2 MOC/WFF Interface**	**100d**	**03/11/96**	**07/30/96**	
8	1.1.2.1 Define and Determine Comm Link LaRC to WFF	2w	04/01/96	04/12/96	AS
9	1.1.2.2 Define Acquisition Data Requirements	1w	04/22/96	04/26/96	AS,SN[0.2]
10	1.1.2.3 Define SAGE Raw Data and 9C Formats	3w	03/11/96	03/29/96	SN[0.2]
11	1.1.2.4 Data Transfer from WFF to LaRC	4w	04/29/96	05/24/96	SN[0.1],AS
12	1.1.2.5 Develop Interface Documents	4w	06/25/96	07/23/96	AS,MC[0.25]
13	1.1.2.8 Schedule Passes	4w	05/28/96	06/24/96	SN[0.2],AS
14	1.1.2.9 Initial Data Transfer Tests LaRC to WFF	1w	07/24/96	07/30/96	AS,MC[0.2]
15	**1.1.3 MOC / FDF Interface**	**155d**	**03/11/96**	**10/17/96**	
16	**1.1.3.1 I2RU AOS/COS**	**20d**	**03/25/96**	**04/19/96**	
17	1.1.3.1.1 Define Station Predict Format	2w	03/25/96	04/05/96	MB[0.1],MC[0.1]
18	1.1.3.1.2 Define IIRV File Format	2w	04/08/96	04/19/96	SN[0.1],AS[0.1]
19	**1.1.3.2 Definitive Orbit**	**65d**	**03/11/96**	**06/10/96**	
20	1.1.3.2.1 Define QA Parameter and Format	2w	03/11/96	03/22/96	MB[0.1],MC[0.1]
21	1.1.3.2.2 Define GPS /Glonass State Vector file Format	1w	04/22/96	04/26/96	MC[0.1],MB[0.1]
22	1.1.3.2.3 Define State Vector Set file Format	2w	04/29/96	05/10/96	MB,MC[0.1],MB[0.1]
23	1.1.3.2.4 Define Job Control Parameter for Flight Dynamics	2w	05/13/96	05/24/96	MB,MC[0.1]
24	1.1.3.2.5 Define Predicted Ephemeris File Format	2w	05/28/96	06/10/96	MB,EP,MC[0.1]
25	1.1.3.6 Prepare FDD SAGE III Operations Guide	2w	06/11/96	06/24/96	MB,MC[0.1]
26	**1.1.3.7 Flight Dynamics**	**80d**	**06/25/96**	**10/17/96**	
27	1.1.3.7.1 Develop Flight Dynamics	8w	06/25/96	08/20/96	MB
28	1.1.3.7.2 Verify, Validate Flight dynamics	4w	08/21/96	09/18/96	SN[0.1],MB
29	1.1.3.7.3 Port Flight Dynamics	4w	09/19/96	10/17/96	SN[0.5],MB
30	**1.1.4 MOC Data Processing**	**365d**	**03/11/96**	**08/19/97**	
31	**1.1.4.1 MOC Data Processing**	**365d**	**03/11/96**	**08/19/97**	
32	1.1.4.1.1 Develop High Level Design Requirements for MOC Data	6w	03/11/96	04/19/96	SN
33	1.1.4.1.2 Design Level 0 Data Ingest	1w	07/16/97	07/22/97	SN
34	**1.1.4.1.3 Health and Safety**	**80d**	**02/07/97**	**06/02/97**	
35	1.1.4.1.3.1 Design SAGE III Health and Safety Reports	4w	02/07/97	03/07/97	SN
36	1.1.4.1.3.2 Design SAGE III Health and Safety Notification Pro	6w	03/10/97	04/18/97	SN
37	1.1.4.1.3.3. Design SAGEIII Health and Safety Standard Data	4w	04/21/97	05/16/97	SN
38	1.1.4.1.3.4 Design Network Link Monitor	2w	05/19/97	06/02/97	SN
39	**1.1.4.1.4. Level 0 Data Collection**	**170d**	**06/04/96**	**02/06/97**	
40	1.1.4.1.4.1 Design and Develop Level 0 Meta Data (DAAC)	1w	09/19/96	09/25/96	SN
41	1.1.4.1.4.2 Design Modal Files	4w	01/09/97	02/06/97	SN,BC
42	1.1.4.1.4.3 Design Level 0 Data (DAAC)	8w	07/24/96	09/18/96	SN
43	1.1.4.1.4.4 Design Data Archive System	2w	12/24/96	01/08/97	SN
44	1.1.4.1.4.5 Develop Data Delivery Method to DAAC/SCF	2w	12/03/96	12/16/96	SN
45	1.1.4.1.4.6 Design and Develop SAGE III Definitive Orbit Form	8w	09/26/96	11/22/96	SN
46	1.1.4.1.4.7 Design Calculation of I2RV	1w	12/17/96	12/23/96	SN
47	1.1.4.1.4.8 Design SAGE III First Data time Process	2w	06/04/96	06/17/96	SN
48	1.1.4.1.4.9 Develop Definitive Orbit Meta Data Code	1w	11/25/96	12/02/96	SN
49	1.1.4.1.5 Design Data Cataloging System	4w	06/17/97	07/15/97	SN
50	1.1.4.1.6 Procure TK Hardware	2w	04/22/96	05/03/96	SN,MC,EP
51	1.1.4.1.7 Procure TK Software	3w	05/06/96	05/24/96	SN,MC

Figure 7. A planning print-out showing relationships of tasks.

Project Planning at NASA

by John R. Chiorini

A number of NASA project teams have recently experienced a change in the way in which they have created their project plans. This has been brought about by a fundamental shift in the understanding of the purpose of the planning process.

The traditional view of planning is that the essential end product of the process is a schedule of anticipated events together with a statement of the resources necessary to perform all required work. Such a schedule is best produced by identifying all necessary tasks, their logical dependencies, the estimated duration of each task, and the resources required or to be made available for the performance of each task. While such a view carries the implicit assumption of interdependencies, durations and resources, there is nothing in the end-product statement that validates such an assumption.

Plans allow the simulation of a project. Too often, however, the finished logic network and resulting schedule are viewed as suitable for "what if" games, and future event management is restricted to anticipating risks and managing tasks on the critical path. Because the physical plan is the simulation, this view assumes that such a plan, whether created by a planning department, by the project manager working in isolation, or by a project team working as a whole, is an equally useful product, as long as it is "correct." That is, as long as it represents the future state of the project, the process by which it was created is immaterial.

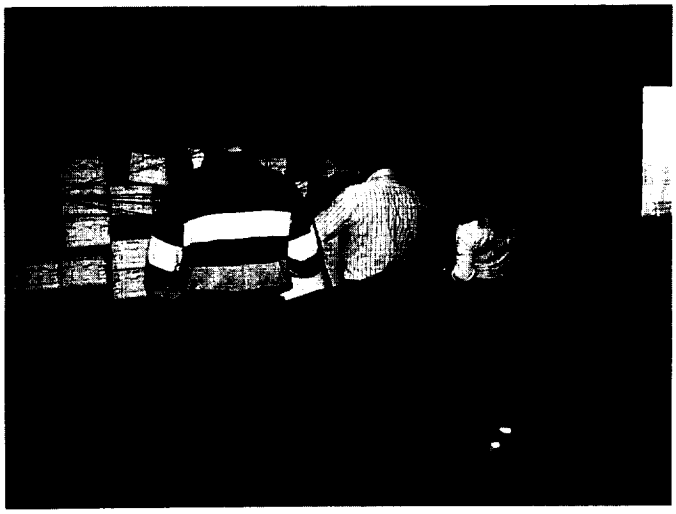


Figure 1. Team Network Development.

The fundamental shift in thinking came with the understanding that the true purpose of the planning process is the translation of requirements into agreements to perform the necessary work. The agreements are made by the members of the team tasked with actual work performance. The schedule, with its underlying logic network and task-level resource plans, is an intermediate product. The agreements are derived from the process of creating that network in a team setting. It is this team process which holds the key to effective planning because validity evolves from the collective decisions made by the project team in the process of creating the project plan. The derived logic network and schedule, which are the end products of this simulation, are more valid than any created in isolation by a planner or project manager hoping to anticipate the future decision of the team.

To date, eight NASA teams have been facilitated in the development of their project plans through a task order contract between NASA Headquarters and the Center for Systems Management (CSM). The teams have included, among others, the Gravitational Biology Facility Project, the Transport Research Flight Facility Project, the Advanced General Aviation Transportation Experiment—AGATE, and both the SAGE Instrument Development and SAGE Software Development Projects. The planning sessions are intensive one-week team events which produce resource-loaded schedules. Facilities used for the planning have included the NASA Wallops Island Management Education Center and off-site facilities provided either through CSM at their plan-

ning center in Cupertino, California, or at other off-site locations provided by NASA.

NASA employees who have attended Project Management training courses conducted by CSM are familiar with the planning process taught by them and the use of facilitated Cards-on-the-Wall sessions to capture the team decisions on the Work Breakdown Structure and project plan prior to entry into the planning software of choice (see Figure 1).

For those not familiar with the process, tasks are first described on large cards (see Figure 2) by team members. Each card contains space to document certain background information on a task, describe the work to be involved in performing the task, identify the input information required to start the task, list output products of the task, describe the estimated duration of the task, and the resources estimated to be needed to accomplish the task work. Team members construct a Work Breakdown Structure using the cards and then link the cards on a large wall with yarn, review the resulting task descriptions and logic with the project manager and the entire team, and only when concurrence is reached, the task cards and logic are captured in project planning software. Because the team gets to participate in the actual planning process, agreements on task interactions, resource commitments, risk mitigation actions, and concessions on durations and hand-off logic are made by the team during the planning process. The initial simulation of the project occurs during the planning, not as some post-plan creation of the logic. That is, the planning process, conducted in a team setting, allows decisions on future events, compromises to be made now that will be implemented some time in the future, workarounds to be planned today to be used, if necessary, at some future event, and agreements to be exercised in the future to hand off products in specific formats to subsequent task teams.

The strength of the process is best understood in the observation from one participant who noted that only

Task Planning Form WBS No: _____

Task Name: _____

Task ID: _____ Estimated Duration: _____ *Circle one* Minutes Hours Days Weeks Months

Task Manager: _____

Form Prepared By: _____ Form Preparation Date: _____

Constraint, Start: _____ Finish: _____

Input	Task Description	Output
1. _____ From:		1. _____ To:
2. _____ From:		2. _____ To:
3. _____ From:		3. _____ To:

Resource Requirements: _____

Figure 2. The Task Planning Form.

one or two people can stand around a 19-inch computer monitor and critique planning logic, but the whole team can stand around the 8-foot by 30-foot wall of the planning room and participate in the creation of project logic.

The facilitation model used for planning by CSM involves a multi-step process carried out over a four- to five-day period.

The team gathers, typically the evening before the actual planning begins. For the best planning, the attendees should consist of representatives of all stakeholders: NASA staff, contractors and their subcontractors if the latter groups have already been chosen. Participants should be able to commit their respective organizations in terms of resources to be expended on tasks and risk mitigation actions. It is essential that all involved stakeholder groups be represented during that opening session and throughout the planning session so that critical decisions about tasks, actions, resources, etc., can be made by the group during the planning session and not deferred to players not present during the actual planning. The first session is an opportunity for introductions and for the project manager to brief the group of the current status of the project, get consensus on any deliverables, and review the work breakdown structure and other planning documents that currently exist. The evening overview is essential to ensure a common frame of reference for all participants.

As a conclusion to the evening, the facilitator then presents an overview of the planning process and explains the work to be undertaken in the next few days. One of the most important discussion points is the definition of the agreed-to event that will constitute the terminal event of the planning: launch, delivery to KSC, etc. All participants must understand the deliverables due at this event so that the deliverables, can be defined in the actual planning process. Also explained in this introductory session are the ground rules by which configuration management will be maintained. The essential ingredient in that process is the role of the project manager as the final arbiter of the information to be entered into the computer after posting on the walls of the planning center.

The planning work begins with the development of a product-oriented WBS or the critique of the current WBS if a suitable product-oriented WBS already exists. The planning cards are used to describe the lowest level of the WBS—task work, and any higher level integration/testing/procurement work. In this way, those cards can be used directly in the creation of the logic network.

Once a WBS has been created and approved by the project manager, a milestone spine is created and placed on the walls. This spine consists of the major milestones for the project, as agreed to by all participants. A milestone is a decision point where progress on some portion of the project or with the project as a whole can be reviewed and approved. For each milestone, participants must agree on the products to be reviewed, the name or office of the reviewer with authority to approve or limit progression, and the nature of the proof to be demanded at the milestone of the readiness to proceed with the rest of the project. The milestone spine provides a physical frame of reference for all participants, indicating points on the planning wall where strings of project logic need to come together. It constitutes a top-level picture of the completed logic network.

Once the milestone spine is created, sub-teams are designated to work on the portions of the logic between network milestones.

Now the logic network can be created with the planning cards connected by yarn to create the physical network. Each card contains a description of the work to be done for a given task, the input(s) needed to start the task, the output product(s), the resources required to perform the work, and the amount of resources needed and/or the duration that those resources will be required.

Once the collective effort of the team has created the network and the project manager has "walked the walls" to review and approve all cards and logic, the data is captured in whatever software the team will be using to manage the project logic once they return home. A critical path is calculated and the team as a whole analyzes the results to determine if the derived dates for milestones meet target dates imposed by

users, launch dates, etc. The process of analyzing the network and shortening the critical path begins by identifying the earliest milestone date that the team judges to be unacceptable. Decisions are made to change logical relationships, reduce durations by adding resources, etc., until the derived date is as close as possible to the team's target date. The next milestone in chronological turn is then analyzed, and so on until dates are accepted for all milestones. This process of network analysis produces the baselined schedule against which the team agrees to work.

Throughout the planning process, five additional activities of major importance to the usefulness of the final product are occurring. All acronyms used in the planning process are listed as the start of a common project vocabulary. Any project risks identified during the planning are listed for later analysis and development of mitigation plans. Any assumptions made during the planning process are listed, as are action items taken by specific team members. Finally, team building is an ongoing activity.

Participants in this facilitated process have universally praised it for its value in bringing the team together and making clear to all team members the interdependencies that exist. To quote a few:

> "It brought all of us together . . . It made us think about the work involved, the chain of action, the flow, the team work, the communication."
>
> "[It] forced me to think through all of the functions that I will have to perform."
>
> "[I particularly liked] the schedule resolution with all interested parties present."
>
> "[It] gave me a scope of the program that I did not have before."

As noted above, one end-product of the planning session is a resources-loaded project logic network with the critical path clearly identified. Sufficient time is always allowed to balance the critical path such that the team can see the actions necessary to achieve target milestones. Perhaps more importantly, another end-product is a clear understanding on the part of the entire team of their mutual interdependencies. The process of creating the project logic network and reconciling scheduling problems builds teamwork and ownership from participants to the shared challenges of completing the project according to the schedule they have produced as a team.

Successful facilitation will require that the team be prepared to dedicate four to five days to this process, and all critical team members must plan to be present for the full planning event. The project's deliverables and internal products must be well-defined and a terminal event must be defined or definable. An existing product-oriented WBS is desirable since, in the absence of one agreed to in advance by the team, one must be created during the planning session. The project team must include one person knowledgeable in the use of the planning software to be used to capture the logic network so that a team member can take responsibility for exercising the software when the team returns to its home facility. Teams are also responsible for providing their own copy of the software to be used, a suitable computer, and a high-speed printer or plotter. If the team desires to resource-load the network, an agreed-to list of resources by name or labor category must be provided or definable during the planning event. Any limitations on the use of specific resources (i.e., limited numbers of a specific resources, limited availability of a specific resource, etc.) must also be known at the time of planning.

If the team proposes to use a facility other than the CSM planning facility or one provided by NASA Headquarters, the facility must include at least 120 linear feet of hard-surface walls on which cards and yarn may either be taped or tacked. The planning room must be dedicated to the process so that the logic network can remain up on the wall throughout the full planning session.

Planning is most effective when it is done as the initial event on a project. Planning must also be done at the transition from one project phase to another or whenever the current state of the project is such that the existing plan is no longer valid because of project changes or discovery that the original plan was an inadequate reflection of the actual project.

The "One-Pager": Methodology & Application, Experiences and Lessons Learned

by Tony E. Schoenfelder and James Wilcox

An article entitled, "'The One-Pager': Methodology & Application" appeared in the Spring 1995 (Volume 9) issue of this publication. The methodology of the One-Pager technique was described in some detail, as were applications in assessing a program's baseline plan and determining progress against the plan. This article will describe the application of the One-Pager in assessing planning alternatives, and will also share some experiences and lessons learned since early 1995. Although a careful review of the previous article would greatly assist the reader in deriving the maximum benefit from this article, the following excerpts will serve to recapitulate the objectives of the One-Pager technique:

- NASA program and project managers need a system that will facilitate timely, accurate top-down program/project assessments required to establish and/or assess the program's baseline plan, determine progress against the plan and assess planning alternatives.

- Cost, schedule and performance measurement systems must operate effectively and efficiently under constantly changing conditions. Existing NASA systems often fail to satisfy these requirements.

- Scheduling and performance measurement systems are often very detailed and generate vast amounts of data, but rarely in a form or format that is conducive to providing timely visibility into today's programs.

- Contractual arrangements between NASA and its contractors do not incentivize the contractors to provide good long-range schedule and cost planning.

- The One-Pager is a single chart that presents an integrated cost, schedule and content (metrics) display for a selected end item. The selection of candidates for One-Pagers is based on the principle that management attention should be focused on major drivers, i.e., those definitive end-items that exhibit one or more of the following characteristics: 1) high cost, 2) high technical risk, 3) high schedule risk, and 4) key integration intersection. There is generally a high correlation between risk (technical and schedule) and cost.

- Who performs the work has no bearing upon whether a system or subsystem is selected for a One-Pager.

- Deciding what not to include is perhaps the most difficult process. Since the objective is to focus management's attention on major drivers, minor products and processes should be reviewed on an exception basis only, and should not be included in a One-Pager.

Assessing Planning Alternatives

NASA programs and projects currently operate in an environment of increasing volatility and uncertainty. One consequence of this situation is the frequent need to engage in program/project replanning activities. Replans are often necessitated by budget reductions, content changes, unanticipated technical problems, schedule slips, cost overruns, or some unique combination of these events. One-Pagers, by virtue of their basic simplicity, facilitate timely, top-down replanning by capturing the critical elements of the project and providing a macro look at the programmatic impact of various changes.

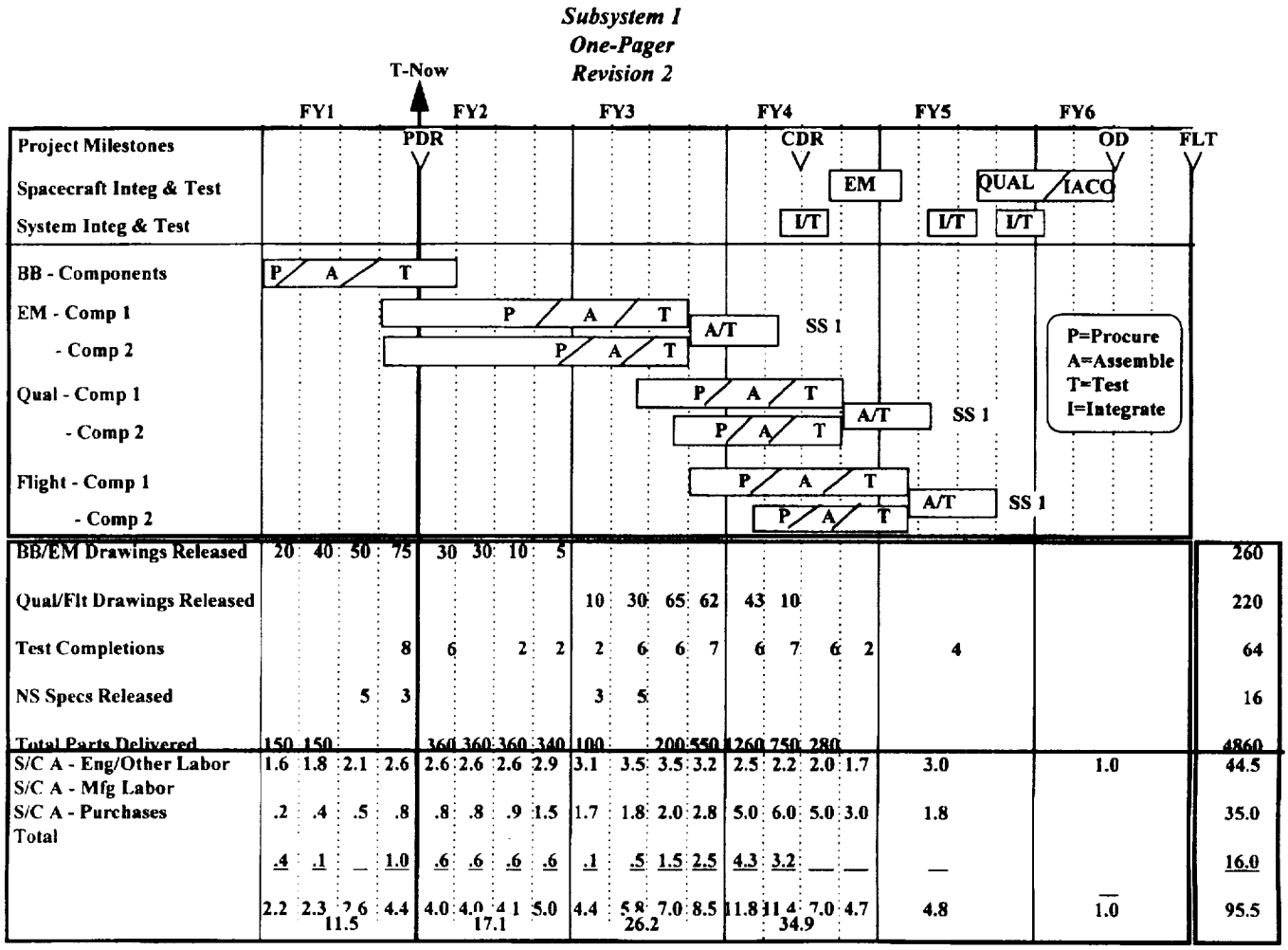


Figure 1. A Simple One-Pager.

Figure 1 was used in the previous article and represents a simple One-Pager for a fictitious spacecraft subsystem. We are currently at T-Now and have just completed the project Preliminary Design Review (PDR). Let us suppose that we have just been notified of the following circumstances, and have but a few hours to provide a credible response:

- Due to project-wide budget constraints, funding across the project will be reduced by approximately 20-25% for FY2 and FY3.

- At the same time, external pressures (from both Congress and our international partners) have dictated that the flight date be given only three months schedule relief.

The first step should be just a simple, overall assessment of the nature and magnitude of the problem and what it implies in terms of any proposed solution. Note in Figure 1 the FY3 4th quarter cost plan for $8.4M (approximately 12% of the $66M FY2/3 spending plan). Clearly, pushing a full three months of costs into the future will not solve the 20-25% reduction requirement, so we must consider other options, such as changes in program logic or content, schedule bar length squeezing and/or slack reduction.

Start first by identifying and considering those actions that can be taken at the project level, where the dollars involved are greater and more responsive to schedule movement. Then consider actions at the sys-

tem and subsystem levels. Our ground rules indicated that we could give the flight date a maximum of three months schedule relief, so our first action should be to move the flight date three months to the right.

Our next action is also at the project level, but its genesis can be traced back to the early stages of creating this One-Pager. Remember that one of the first steps to be taken in building the baseline plan was to review the schedules, understand how they were developed and identify the underlying assumptions with respect to bar length, shifting, lead time, etc. This knowledge would aid in calibrating the overall risk inherent in the schedule rationale, and would identify areas where future actions might be taken. When we reviewed the underlying assumptions of this particular schedule, we learned that the spacecraft integration, assembly and check-out (IACO) was to be performed on a single-shift basis. Notice in the baseline schedule at the top of Figure 2 that the IACO bar length is eight months long. By adding a second shift and utilizing an accepted program analysis rule of thumb that a second shift is approximately 70% as efficient as the first, IACO is reduced to five months (8÷1.7 = 5). This IACO compression, in concert with the three-month slip to the flight date, yields a six-month slip to the start of IACO (See Figure 2, Rev. 1).

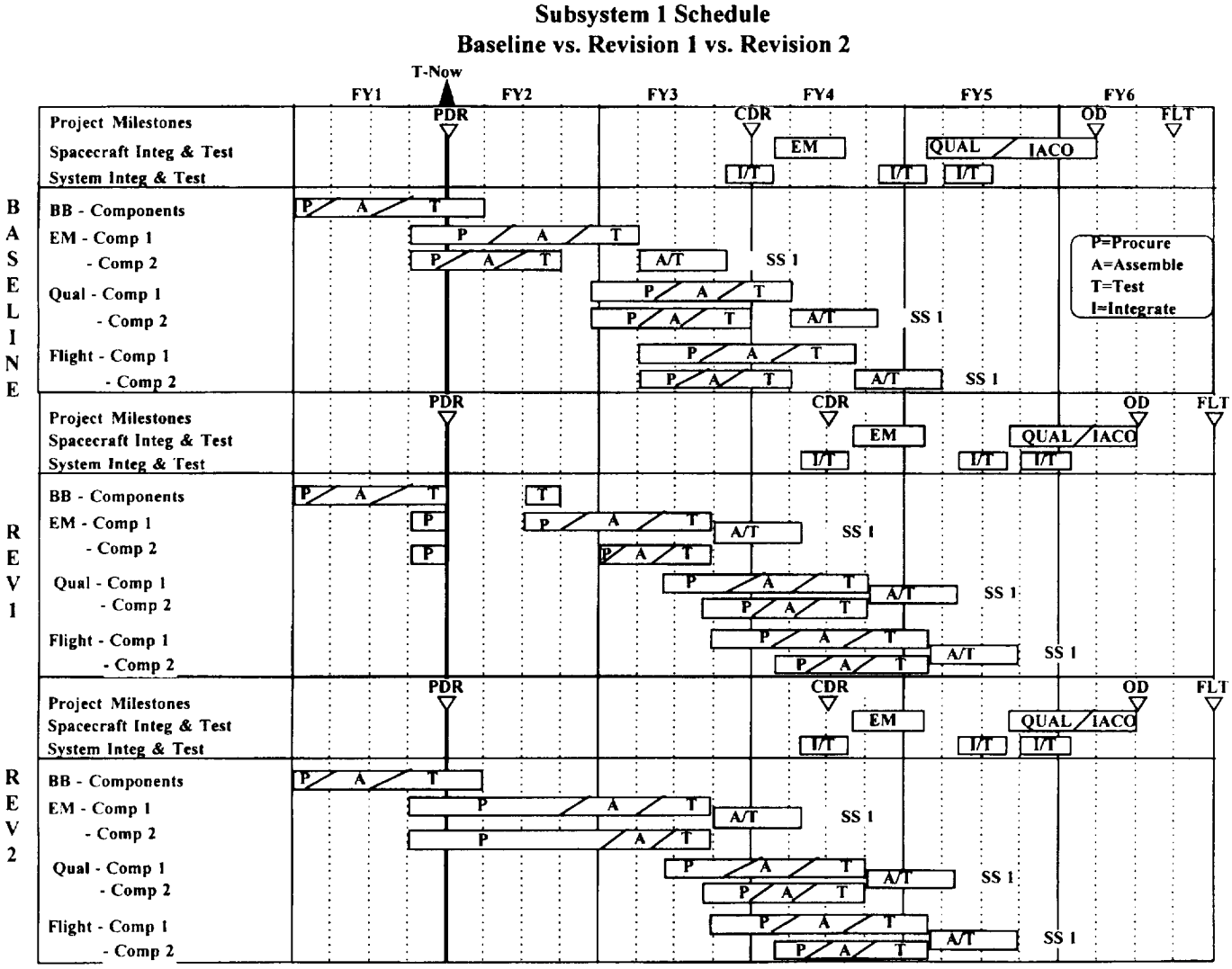


Figure 2. Baseline with two revisions.

The next steps should be taken at the system and subsystem levels. All of the system and subsystem activity bars should be moved six months to the right, retaining the same orientation to one another as in the baseline. No attempt should be made at this time to adjust bar lengths or take any other action which might call into question the validity of the exercise.

Notice in Figure 2, Baseline, that although the procurement activities for the various fidelities of both Components 1 and 2 begin at the same time (probably for the convenience of the procurement process), there is from three to six months' worth of slack between the completion of testing of the various fidelities of Component 2 and the beginning of Subsystem 1 assembly and test. This presents us with yet another opportunity to move scheduled activities. By simply moving the activity bars for the various fidelities of Component 2 to the right until all slack is removed (See Figure 2, Revision 1), we eventually move additional costs out of the constrained years.

Finally, notice in Figure 2, Revision 1, that there is an apparent gap of six months between the T-Now line at PDR and the future scheduled activities. From studying the completed schedule activities and metrics found on Figure 1, we observe the following:

- Subsystem 1 is well into its design phase.
- Roughly 70% of the breadboard/engineering model (BB/EM) drawings have been completed.
- The project PDR has just been completed.
- Specification releases and purchase orders for the engineering model part have been issued.

It would be too disruptive and inefficient to attempt to terminate the project and then restart it six months later. Our final action should be to stretch the engineering model schedule over the six-month gap and work at a lower spending rate (See Figure 2, Rev. 2). This maintains momentum on the breadboard and engineering model units, takes full advantage of relief to both qualification and flight hardware deliveries, and delays the buildup in both the engineering and manufacturing workforces.

The final step is to adjust the costs and the metrics to reflect the revised schedule. Figure 3 shows a One-Pager for Subsystem 1 which reflects all the changes made to accommodate the 20-25% budget reductions in FY2 and FY3.

An experienced analyst can easily adjust the baseline cost plan to both fit the new schedule restraints and provide a smooth transition from T-Now into the replan. An examination of Figure 3 will reveal the following:

- The total Estimate-at-Completion grows from $90M to $95.5M, reflecting a penalty of $5.5M due to schedule stretch and some disruption;
- The FY2 Engineering spending rate avoids the immediate FY2 build-up, while the peak activity moves into FY3. The brunt of the penalty falls in the Engineering/Other category;
- The Manufacturing spending rate avoids a build-up until FY3, and the peak activity moves completely out of the FY2/3 timeframe;
- The Purchasing replan maintains appropriate relationships between spending and scheduled procurement activities.

Figure 3 also shows the adjustments made to the baseline metrics plan to fit the new schedule. An experienced analyst can calculate a revised metrics phasing which retains the baseline metrics/schedule relationships. Note in Figure 3 that the revised metrics plan maintains continuity for engineering drawings and parts deliveries, and previous relationships, such as NS Spec Releases vs. the start of procurement for Qual units, remain in place.

Utilizing the methodology just presented, an experienced analyst could accomplish this replan in a cou-

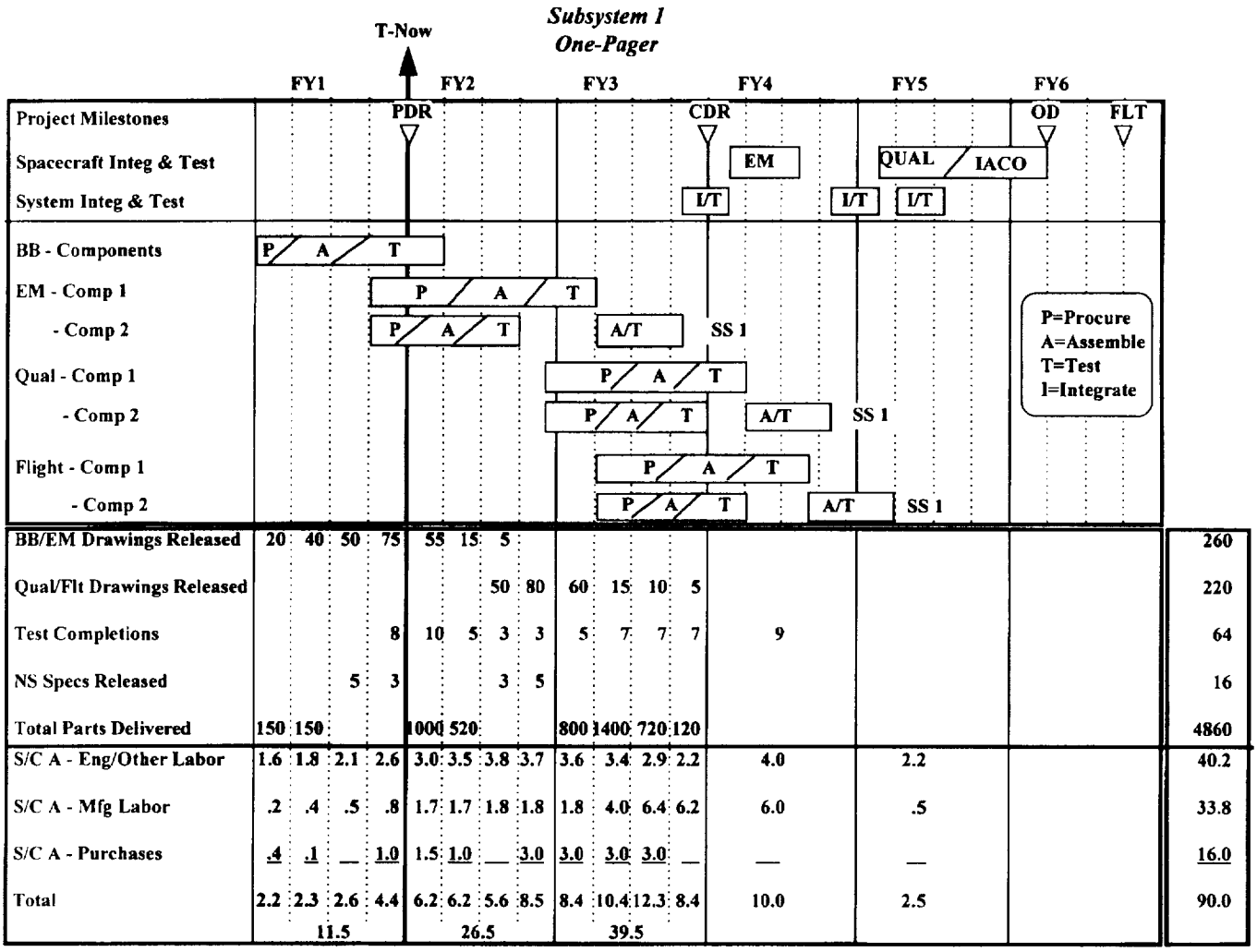


Figure 3. An adjusted One-Pager.

ple of hours, including consultation with a knowledgeable technical person. The accuracy would be entirely sufficient to support management-level decisions.

Experiences and Lessons Learned

Since Spring 1995, considerable effort has been expended in incorporating the One-Pager critical element analysis technique into several large applied technology projects. In addition, the One-Pager technique—whereby a template embodying a discrete set of selection criteria is used to identify activities to be tracked for each critical element—was used to produce an integrated schedule summary for a large spacecraft development project. The following lessons learned are the results of these and other experiences.

Lesson 1

The One-Pager itself has evolved into a One-Pager packet comprising four charts. These charts are, in the order in which they should be developed:

Step 1. Summary Level Logic Network

Step 2. Logic Network Description

Step 3. All-Year Cost, Schedule & Metrics

Step 4. Near-Term Cost, Schedule & Metrics

Step 1, developing the summary-level logic network, has proven to be the most difficult yet most important step toward successful implementation of the One-Pager approach. When the precursor to the One-Pager was developed, the originators of the technique were working in a large development project where the overall logic was identified and well understood. What they did not fully appreciate was that at the inception of a project, logic is developed from the top down, and is relatively simple and well understood by many. However, over an amazingly short period of time, as the major parts of a project are dispersed to different contractors and subcontractors, the overall logic flow becomes more complex, convoluted, and understood by only a few.

Developing the summary logic network as the first step in implementing the One-Pager approach enables all the project participants to see exactly how the major pieces fit together and relate to one another.

Project logic should be established from project inception through project completion, and should clearly and concisely outline how the project will converge on the final product. Figure 4 illustrates the relationships of design cycles, test cycles and project milestones for both a spacecraft development (Phase C/D) project and a pre-Phase C/D applied technology project. Note that both projects converge in the same manner, and that the same techniques can be applied to both. Note also that in both cases, the test programs related to each design cycle are the most concrete and easily communicated measure of the project plan, and thus should be highlighted in developing the summary project logic.

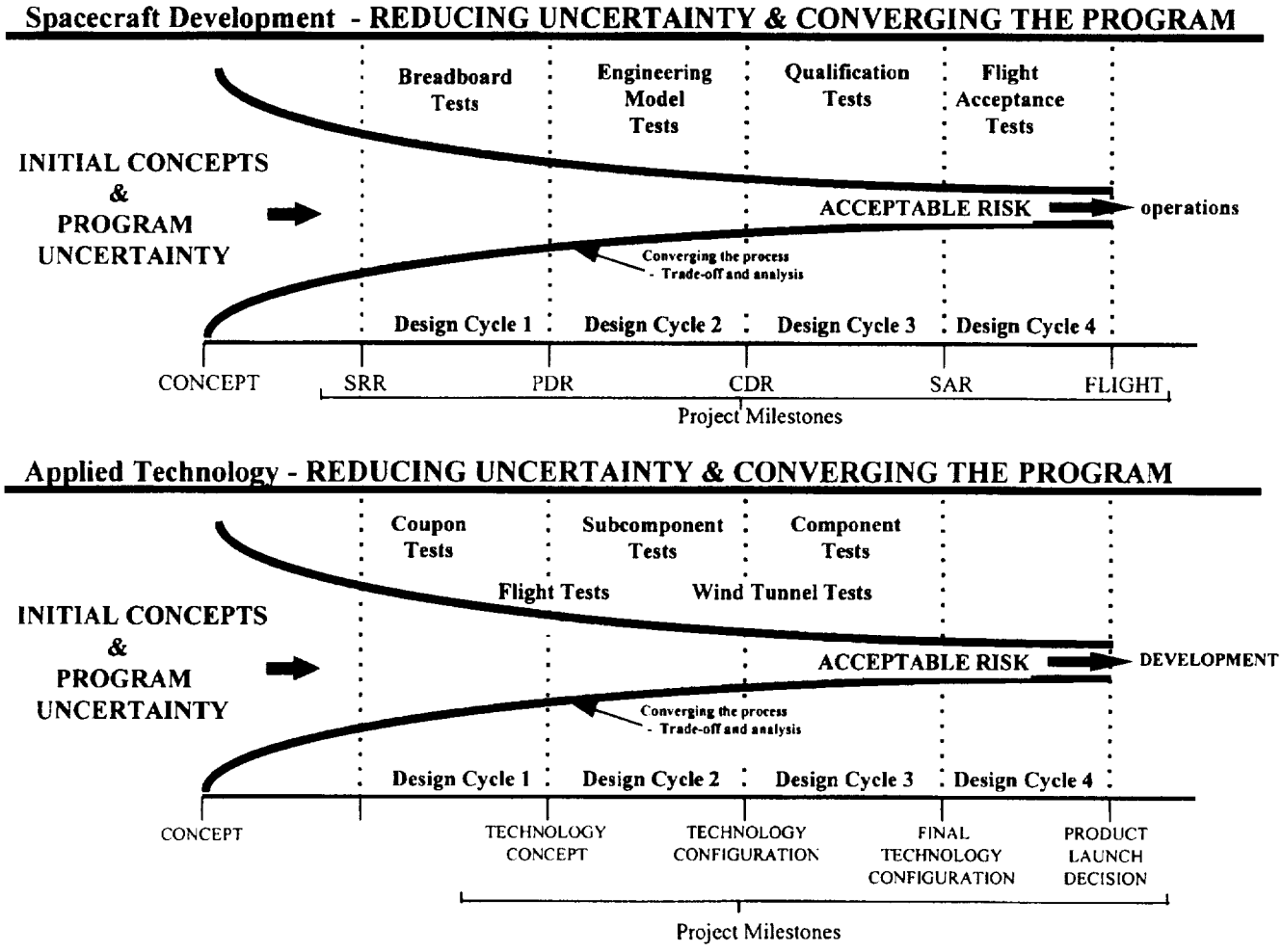


Figure 4. Project Logic.

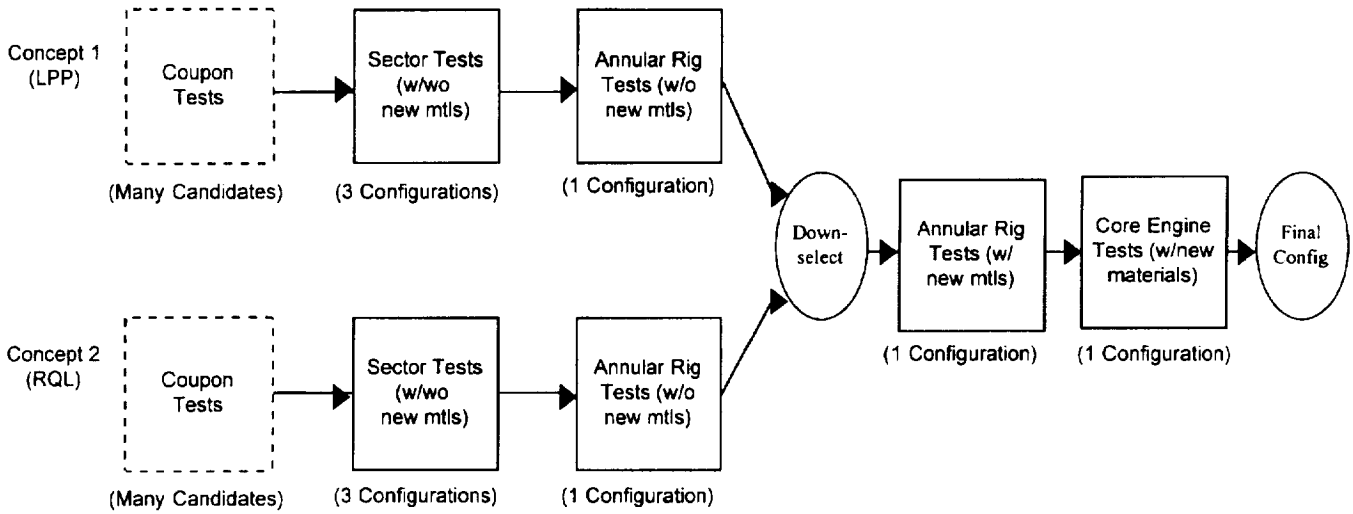


Figure 5. Logic Network.

Figure 5 displays a very top-level view of the logic network of an applied technology research project for an improved combustor. With only a cursory review, one can rapidly observe the following:

1. Two competing concepts will undergo the following test cycles:
   - Coupon testing
   - Sector testing with and without new materials
   - Annular rig testing without new materials
2. Following the test cycles and core combustor design, a downselect will occur.
3. The selected concept will then undergo the following test cycles:
   - Annular rig testing, with new materials
   - Core combustor testing
4. The final test cycles will validate that the concept is ready for the development phase.

Figure 6 is a slightly expanded version of this logic network at about the right level for One-Pager purposes. Each logic box or node is identified by a WBS-like number, the importance of which will become readily apparent.

Step 2, developing the logic network description, requires the identification of the key features of each box, including the products entering and leaving, the activities and/or tests performed there, and any other useful information concerning that box. Notice in Figure 7 that each logic description has a number that corresponds to a logic box found in Figure 6. By referencing the logic box number and consulting the associated logic description, it is possible to immediately find out what is occurring there. Notice also that special attention is devoted to describing the number of candidates tested in each cycle, the nature of the test programs, and the relationship of one logic box to others.

In addition to providing increased visibility and understanding, these summary logic networks have been shown to be excellent aids to communication. Discussions concerning some aspect of a project are considerably enhanced by using the appropriate logic network to provide much-needed context.

Steps 3 & 4, development of the All-Year and Near-Year Cost, Schedule and Metrics charts, were covered in extensive detail in the Spring 1995 issue of this publication. Therefore, no further discussion is offered herein except for the following: The basic

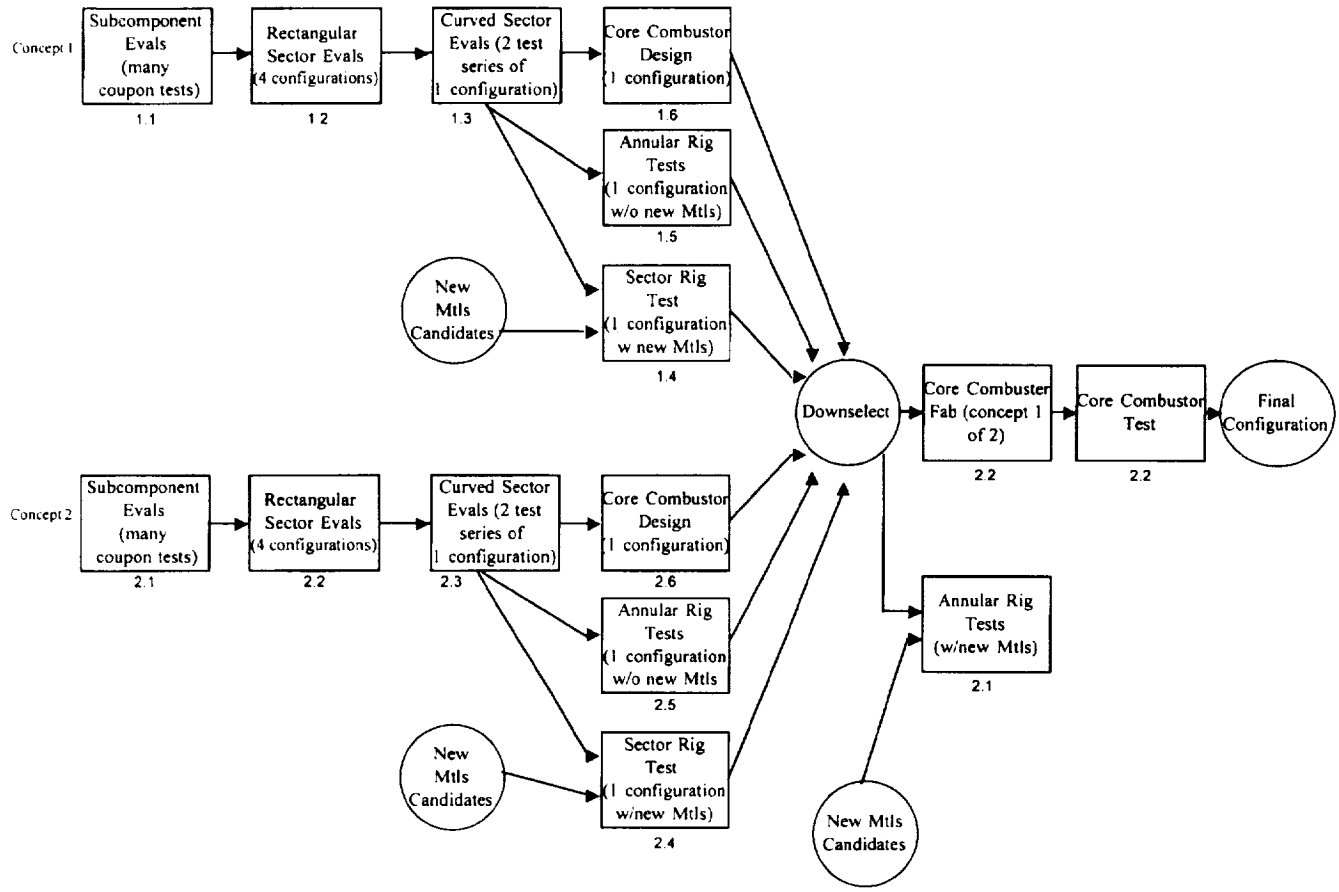


Figure 6. Logic Network expanded.

1.1 & 2.1 Subcomponent Evals
- Many coupons tested
- Feeds sector test program
- Continues during sector test prog
- Used for sector design refinement

1.2 & 2.2 Rectangular Sector Evals
- Combines components for integrated evals
- 4 configurations tested for each concept
- Primary feed to annular test program design
- Secondary feed to core combustor test program design
- Uses no new Mtls

1.3 & 2.3 Curved Sector Evals
- Added shape fidelity over rectangular evals
- Two test series of single configuration for each concept
- Feeds core combustor test program design

1.4 & 2.4 Sector Rig Tests
- Actual liner candidates from New Mtls program added to test configuration
- Feeds downselect decision

1.5 & 2.5 Annular Rig Tests
- Full up combustor components combined
- 1 Configuration tested for each concept
- w/o new materials
- Feeds downselect decisions

1.6 & 2.6 Core Combustor Design
- 1 Configuration for each concept
- Includes engine modification, systems integ & instrumentation design
- Feeds downselect decision

2.1 Annular Rig Tests
- Final liner from New Mtls program added to test configuration
- Feeds core combustor test program

2.2 Core Combustor Tests
- Fab selected combustor concept
- Modify engine
- Includes test prep, core engine assy & instrumentation, test, and data analysis

Figure 7. Logic Descriptions.

One-Pager template for the Cost, Schedule and Metrics chart was originally limited to 20 lines of data. This was a deliberate act with a twofold purpose. First, in an effort to maintain the utility of the chart such that problem areas tended to "jump off the page," it was thought that more than 20 lines of data would present too much clutter. Second, it forced the person preparing the One-Pager to select wisely from among a large body of competing data. Experience has taught that up to thirty lines of data can be incorporated into the One-Pager without destroying its utility. Finally, we would like to emphasize that a good job of preparation in Steps 1 & 2 will make Steps 3 & 4 relatively simple to accomplish. An All-Year Cost, Schedule and Metrics chart is provided for your information in Figure 8. A Near-Year chart contains the same data, but covers only 18 months.

Lesson 2

The logic network you build and the schedules you select should focus on activities leading to a specific convergence or milestone. Activities describe the step-by-step process for arriving at a convergent point, e.g., design, fabrication and test, or design, code and test. By tracking activities, you can observe progress, anticipate problems and take appropriate early corrective action. If you limit your focus to delivery milestones, you will know if a milestone has been met only when the due date arrives. You will not know how well the milestone has been met until it is far too late. A review of one possible scenario of the combustor example illustrates the point (See Figure 9). In this scenario, the baseline plan called for coupon, sector and annular rig tests to be performed prior to downselecting a concept. The actual

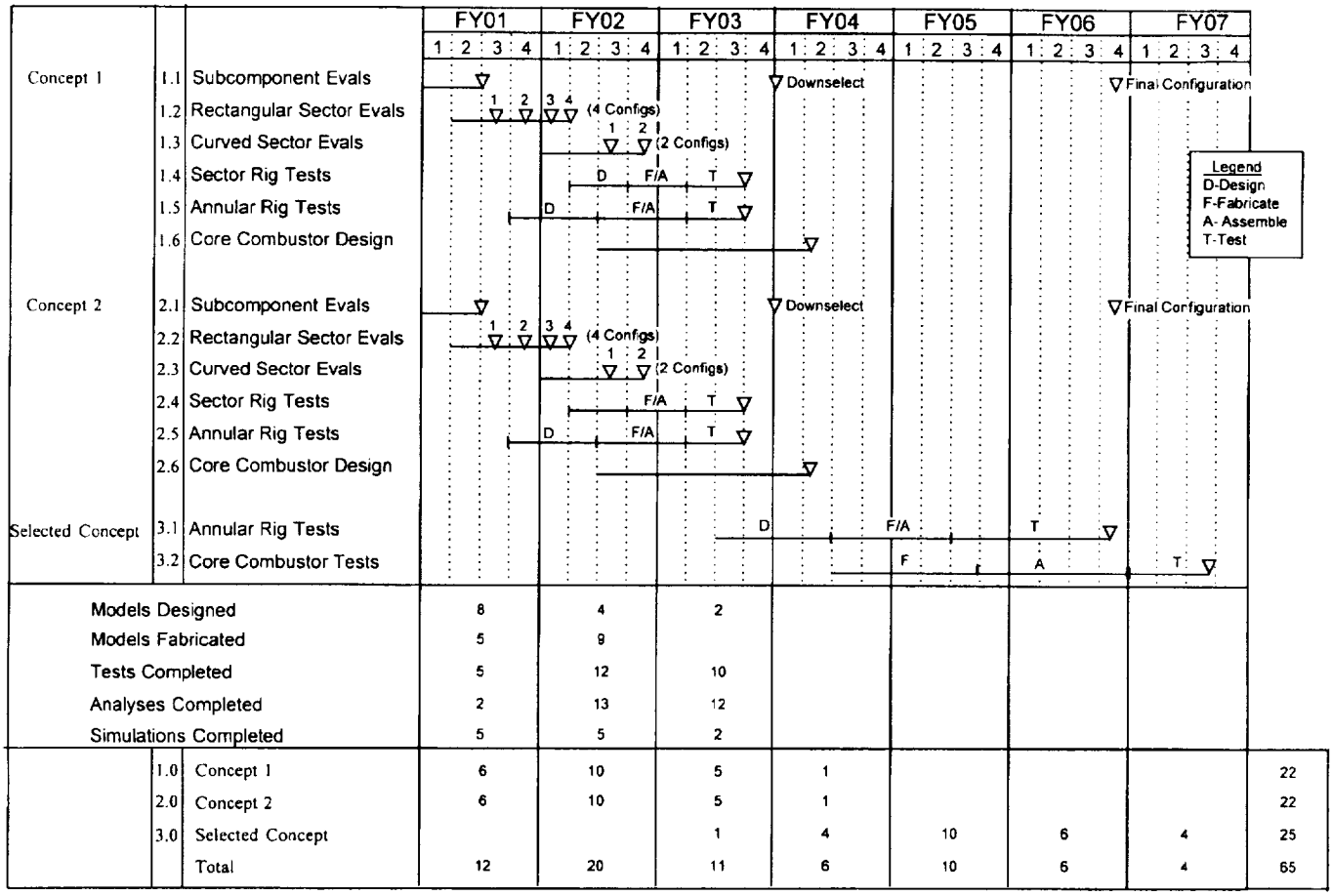


Figure 8. All-Year Cost, Schedule and Metrics.

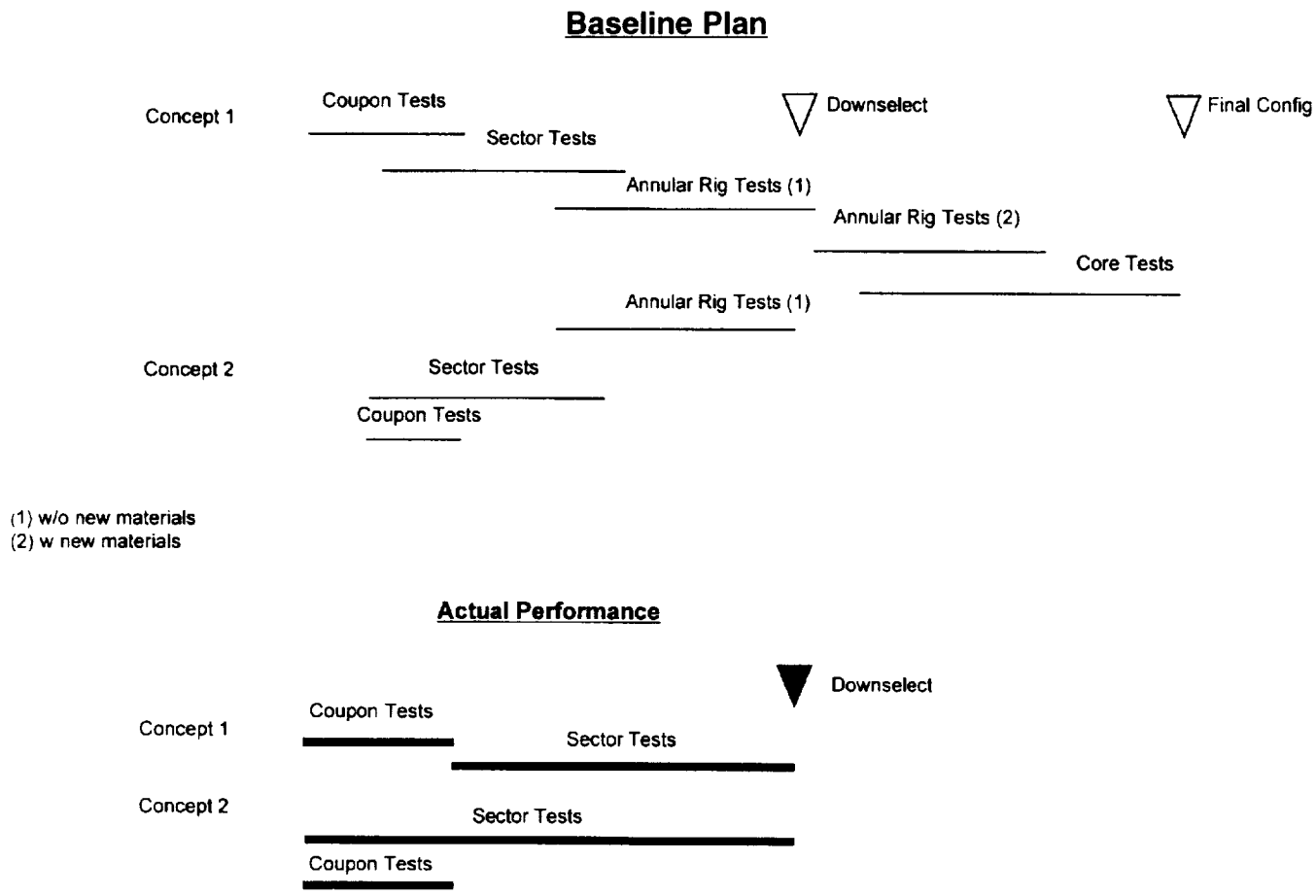


Figure 9. Baseline Plan vs. Actual Performance.

performance shows that the downselect milestone was met; however, the overall quality of the milestone was compromised because the annular rig tests were deferred into the future. Assuming that the dollars originally required to arrive at the compromised milestone were spent, achieving the final configuration milestone will likely require additional dollars and a longer schedule.

Lesson 3

Representatives of various project elements, e.g., IPT's, system and subsystem managers, contractors, etc., may on occasion insist that One-Pagers are of no added value to them and, in fact, intrude upon their autonomy. Accusations of micro-management have, at times, been hurled. If you are attempting to implement a One-Pager correctly, you are actually defining the information you need and the formats you will use at an intermediate level, not a lower level. You will be using existing data, and you do not care how the data is structured or managed below that intermediate level. You must carefully think through this entire issue before implementing a One-Pager, and you must be prepared to deal with some negative feedback. You need to be able to clearly describe what you are trying to accomplish and why. Samples of a completed product may often help to deflect or defuse criticism and turn it into support. Your success also depends upon the degree to which project management is convinced that this is the right way to go and lends its unqualified support.

Lesson 4

If you wait until a stable baseline is in place before you begin using the One-Pager to assess project status and performance, you may never start the

process. Force yourself to start assessing project status and performance, and do not allow yourself to lose this discipline.

There is a need for both near-term and strategic performance measurement, and the two measurements have different objectives. Near-term performance measurement is performed either monthly or quarterly, and seeks to determine progress against the current baseline plan. Strategic performance measurement should be performed annually, and addresses macro performance over a period of at least a year. Strategic performance measurement also looks at the changes in both risk profiles and logic relationships, and seeks to assess their impact on overall program health.

The following example illustrates the dynamic nature of most projects and highlights the different objectives of near-term and strategic performance measurements.

Figure 10 displays a baseline program established at the beginning of FY96. There is an all-year baseline and a more detailed baseline for fiscal year 1996. During the first year, the FY96 baseline was replanned in December and again in March (See Figure 11). The actual cost and schedule status at the end of FY96 is also represented. Using the most current plan (3/96), the computations in Figure 11 suggest that the project should receive a good grade (B+), as the overall accomplishment ratio was .87. This is a perfectly valid measurement and is consistent with the manner in which formal performance measurement systems are supposed to work.

However, a strategic performance measurement taken annually would address the following questions:

- What was the earned value in a macro sense?

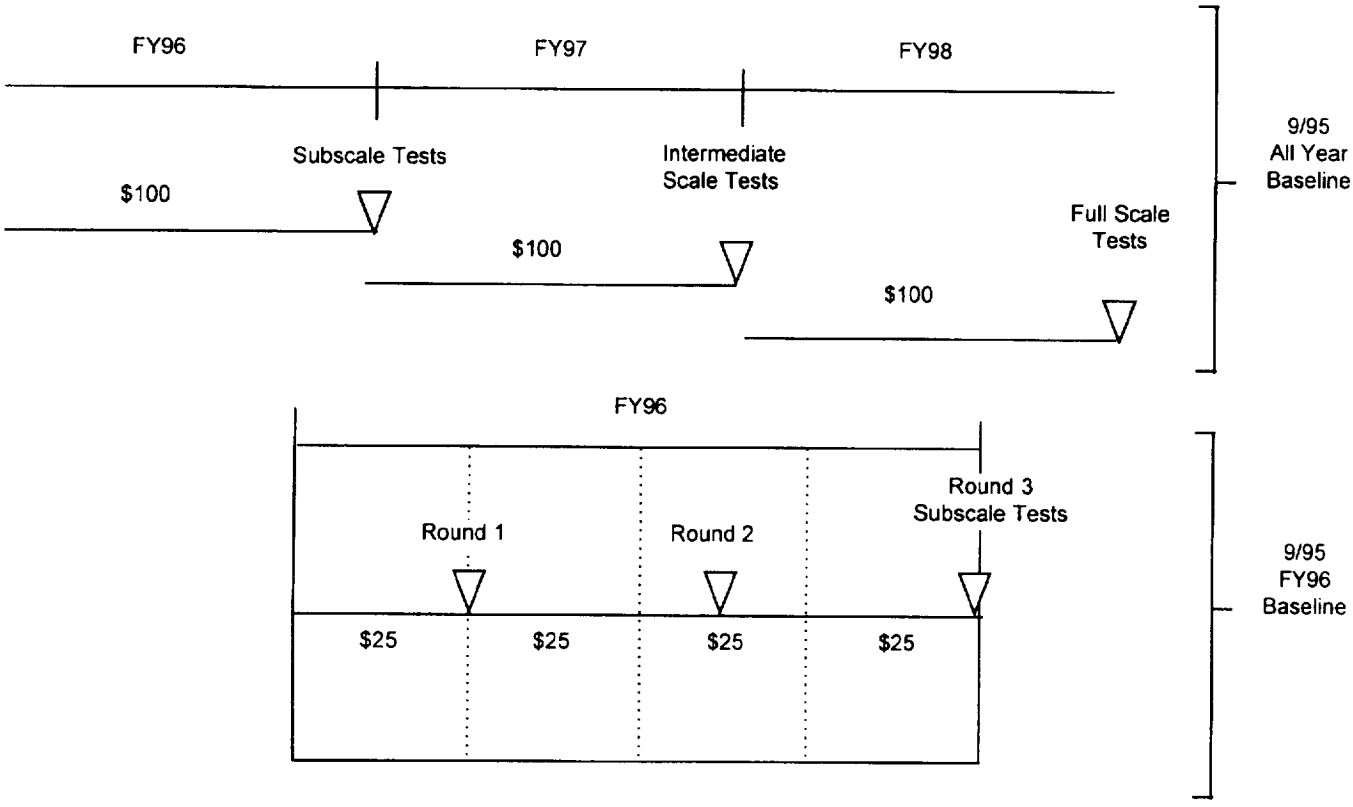
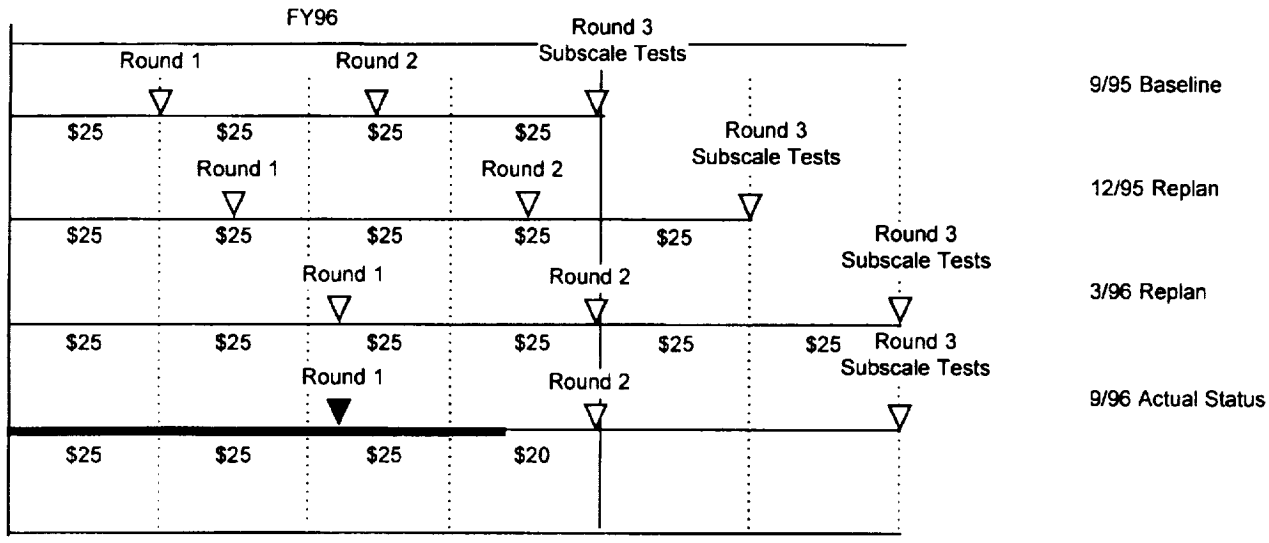


Figure 10. Near-Term and Strategic Performance Measurement.



The earned value computations for the end of FY96 should be based on the most current plan, i.e. the 3/96 plan, and would be computed as follows.

$$\text{Spending Ratio} = \frac{\text{Actual \$ Spent}}{\text{Planned \$}} = \frac{\$95}{\$100} = .95$$

$$\text{Sch Accomplishment Ratio} = \frac{\text{Months Accomplished}}{\text{Months Planned}} = \frac{10}{12} = .83$$

$$\text{Overall Accomplishment Ratio} = \frac{\text{Sch Acc Ratio}}{\text{Spending Ratio}} = \frac{.83}{.95} = .87$$

Figure 11. Earned Value Computations.

- What programmatic objectives have been compromised by accommodating this year's problems?
- Has risk been added to the out-year plan by increasing parallelism and shortening time spans?
- Is the out-year plan still valid and achievable, or have cost and schedule been force fitted to an unachievable plan?

Figure 12 illustrates the strategic measurement of this project. Remember from Figure 11 that the baseline was replanned twice, such that the completion of the Subscale Tests now occurs 18 months from the start of the project rather than the original 12 months. A strategic look at schedule accomplishment at the end of FY96 would indicate that the project has only accomplished 10 months of what is now an 18-month plan, yielding a schedule accomplishment ratio of .56. The resultant macro overall accomplishment ratio of .60 is far removed from the B+ grade computed earlier. It is very important to periodically perform this kind of "conscience" check. Subtle problems can cause a project's schedules to drift to the right, yet the effects of this drift tend to remain undetected by near-term performance measurements, particularly in cases where the baseline is adjusted frequently. By forcing yourself to go through the analysis, you and the rest of the project will be in a position to address the schedule drift factor in a timely manner. Many projects have drifted into severe difficulty because they failed to take this kind of macro view.

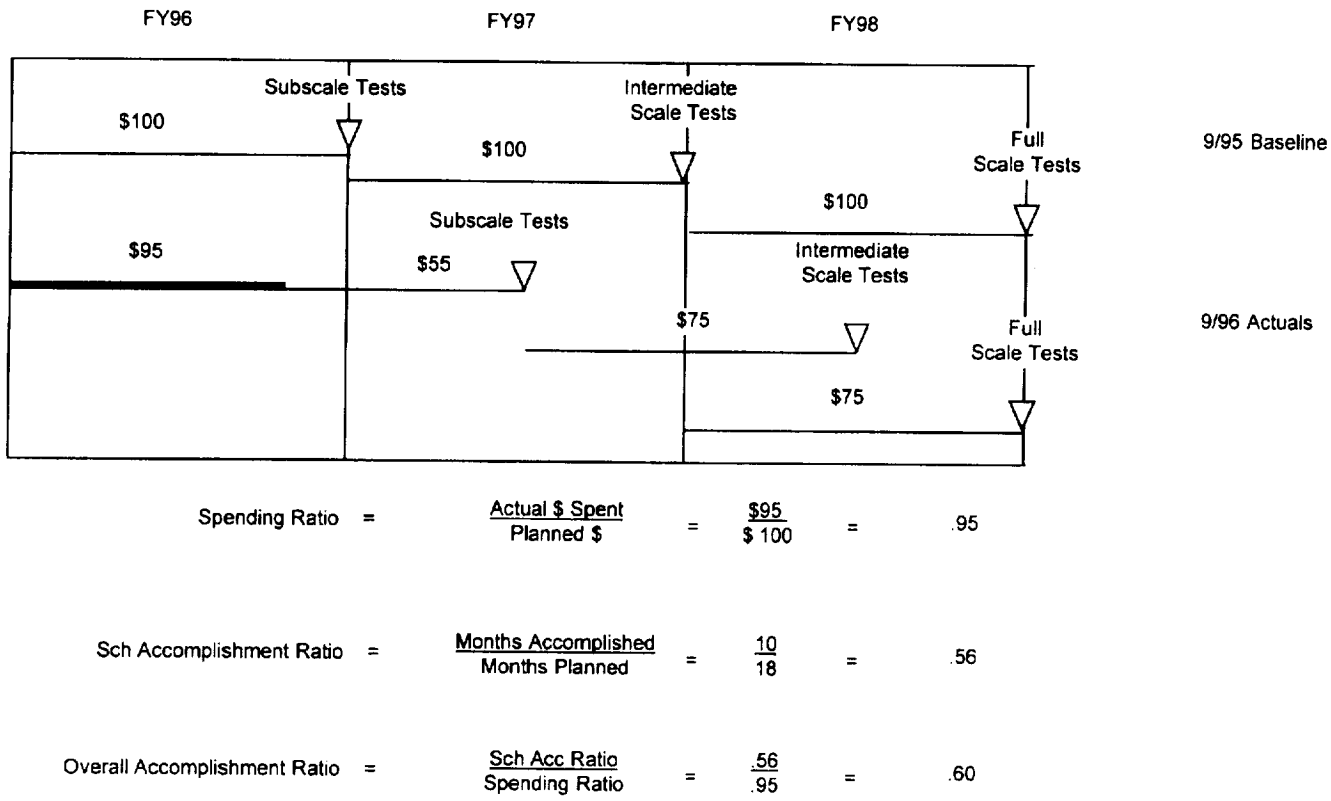


Figure 12. Strategic Measurement.

Lesson 5

It would be wise to solicit help from someone who has prior experience in the execution of the One-Pager process, and it is mandatory that a knowledgeable project office civil servant be dedicated to the task of coordinating the One-Pager development process.

Putting a One-Pager system in place is not easy. It requires first that you understand and accept the philosophy that sometimes "less is more." You must also be able to identify and lay out logic flows, define templates and select appropriate schedule activities, and develop costs and metrics at the proper levels. You must, above all, have a clear vision of your ultimate destination, because you will be plowing through mountains of data in search of the right pieces. Someone who is experienced in this process would prove invaluable, because the exercise is quite different from anything most projects have done before.

Particularly during the early phases of establishing a One-Pager system, there is a great deal of coordination required. The right people must be made available at the right time, and encouraged to cooperate to the fullest. There must be a dedicated civil servant who has both the knowledge and the authority to ensure that the proper degree of cooperation and coordination occurs. Without this person, success will be difficult, if not impossible, to achieve.

Lesson 6

If your program/project is large, with many systems and/or subsystems, you might want to consider an additional step to help focus attention on the major drivers, i.e., those definitive end-items that exhibit one or more of the following characteristics:

1. High cost
2. High technical risk

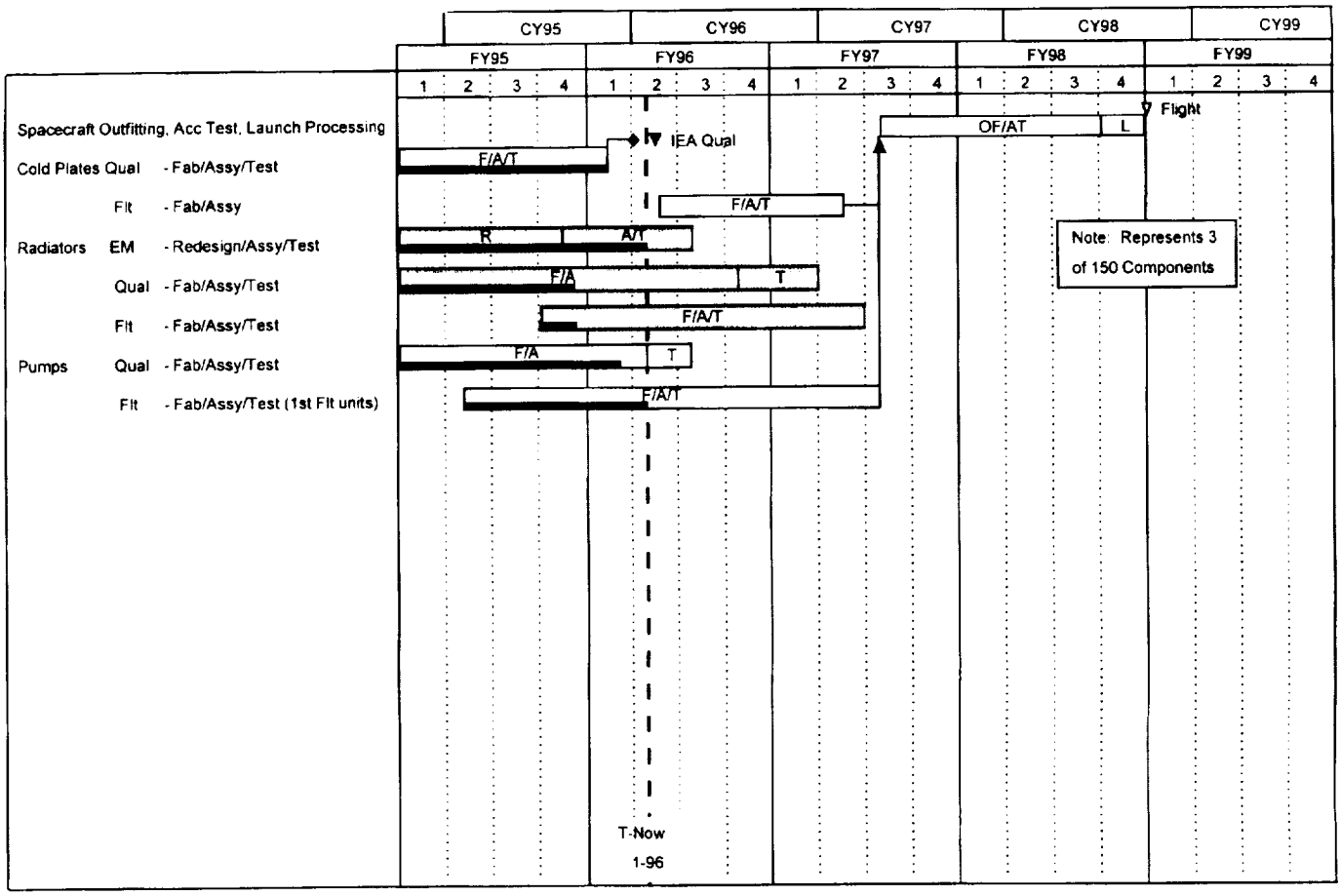


Figure 13. Thermal Control System.

3. High schedule risk

4. Key integration intersection

In a large program such as the Space Station development program, there may be as many as 1,000 major definitive end-items in the program. Using the One-Pager technique, you may have reduced the focus list to 150 end-items. A further narrowing of focus may be achieved by using standard risk criteria to rank each of the 150 end-items. Those items which receive high scores are singled out for special management attention in the normal course of providing program/project status and performance measurement. Figure 13 shows an example of a One-Pager-type schedule for a thermal control system. The radiator activity bars are darkened to indicate that they are critical path items. The heavy black line indicates progress as of T-Now.

Figure 14 shows the Critical Path Survey form with the six standard criteria. These criteria have been used to assess the risk in the ATCS radiator's path.

Experience has shown that, with the assistance of knowledgeable project personnel, a critical path survey can be done in relative short order with dependable results. The following is a brief discussion of what one should consider for each criterion:

- Design Difficulty
  - Has performance has been scaled up from a lesser design?
  - Are there complex or critical interfaces? If so, are there many?
  - Will this design have to satisfy a number of different users?
  - Is new technology required or involved in the design?

Item: ATCS - Radiators

Subsystem/Element Manager: John Jones

Months to 1st Flt Delivery: 15 (T-Now = 11/95)

Months to 1st Flt Nee: 15

Factor	Description	High Risk 3	Moderate Risk 2	Low Risk 1	Remarks
1. Design Difficulty	Significant increase in performance requirements of an existing technology and/or complex interfaces	Significant	(Moderate)	Little	Deployment mech, fluid lines
2. Historical Problem Area	Degree of past cost, technical or schedule	(Significant)	Moderate	Little	Large cost growth, schedule drift
3. Development Maturity	Degree of develoment program maturity vs flight delivery	(Little)	Moderate	Significant	Dev. model redesign; Tight Qual/Flt relationship
4. Status	Behind current schedule plan	Significant	(Moderate)	Little	Qual assy 3 months behind schedule
5. Workarounds	Relief available from extra shifts or alternate	(Little)	Moderate	Significant	Facility constraints
6. Slack	Time between need date and planned completion	(No or Negative)	Moderate	Significant	No planned slack

(Circle column 1, 2 or 3 for each factor)

Average Score: 2.6

Figure 14. Critical Path Survey.

- Historical Problem Area
  - To what degree have cost, schedule and/or technical problems occurred in the past? Is there a history of cost overruns, schedule drifts or requirements changes?
  - What is the performance capability of the contractor? Is this the A-Team? Is there a broad experience base?

- Development Maturity
  - How much parallelism is there with respect to engineering models, qual units, and flight hardware?
  - Is there a modified development template, such as protoflighting?
  - How does the build span (# of months) compare with hardware of similar type and complexity?

- Status - What is the actual schedule performance to date vs the current plan?
  - Risk ranking of 1 = low = 0 to 1 mos. behind
  - Risk ranking of 2 = mod = 2 to 3 mos. behind
  - Risk ranking of 3 = high = 4+ mos behind

- Workarounds - Are workarounds possible due to the availability of some or all of the following?
  - Additional shifts
  - Alternate logic
  - Schedule compression
  - Additional equipment or skills

- Slack - Does the planned completion date support the planned need date?
  - Risk ranking of 3 = high = 0 to 2 mos slack
  - Risk ranking of 2 = mod = 3 to 6 mos slack
  - Risk ranking of 1 = low = 7+ mos slack

A note of caution is in order: After you have obtained inputs from your various project sources, and before you assign final values to the different risk criteria, you must do a bit of reconciliation. For example, a structures engineer may rank the risk associated with the new design of a particular structure as high. Yet

when compared to the high risk associated with the design of a new piece of complex avionics requiring new technology, the structures risk would not be of equal footing. You need to be the final arbiter to ensure that the final risk rankings of the various critical paths are balanced with respect to one another.

Please remember that warning signals do not always flow up to the project manager early enough to permit the most effective corrective action. In many cases, the contractor is incentivized to view the future in a dangerously optimistic fashion. It is up to you to establish the protocols to flush out problems in a timely manner. The small investment required of an approach like this will force improved communications and aid in setting the right agendas. On smaller projects, the project manager may do this kind of ranking in his or her mind, however, as the size and complexity of a project grow, the ability to comparatively analyze all components becomes virtually impossible without a communication aid of this type.

The One-Pager critical element analysis technique results in a packet comprising four charts for a selected end item:

1. Summary Level Logic Network
2. Logic Network Description
3. All-Year Integrated Cost, Schedule and Metrics display
4. Near-Term Integrated Cost, Schedule and Metrics display

The technique was designed to help management focus on key cost, schedule and technical drivers and serve as a common basis for communications. The products are simple in concept and appearance, are produced using a consistent methodology, focus at the subsystem or key ORU level, are done in the context of a hardware/integration/test "backbone," capture only the important "nuggets," and place the emphasis on "programmatics" (the interplay and relationship between the cost, schedule and technical aspects of a program). The One-Pager is not easy to develop, but is relatively easy to maintain, and once in place, will prove to be a powerful tool that will enable project managers to manage more effectively.

A Project Control Milestone Approach to Schedule Control

by Walt Majerowicz

One of the principal benefits of logic network scheduling is that it provides a mechanism for the project manager to focus on potential schedule problems in order to apply the resources necessary to reduce, mitigate or avoid them. However, logic network diagrams can be cumbersome for the project manager to personally manage from, especially on major projects which consist of hundreds of activities, milestones and interrelationships. Likewise, the various Gantt charts, tabular listings, histograms and other products which today's automated project management systems are capable of generating can be overwhelming. And while a detailed schedule is important, the control process can be augmented through the technique of monitoring Project Control Milestones (PCMs). PCMs enable the project manager to understand the schedule "big picture" and focus on urgent schedule issues with the confidence that the PCMs are supported by the underlying detail contained in an integrated project logic network.

The first step in using the PCM approach to schedule control is identifying a suitable set of PCMs. A milestone is an event which represents the start or completion of an activity and is based on a fixed point in time. In general, milestones fall into three categories: major, contract and detail. A major milestone is as its name implies: a key event or one of extremely high visibility such as a Critical Design Reviews (CDR) or launch date. Contract milestones are those in which a supplier is legally obligated to deliver a product or service on a specified date. While major milestones can also be contract milestones, other examples of contract milestones are delivery of a hardware component, completion of a first article qualification test, delivery of a technical data package or completion of a facility's construction. Finally, detail milestones represent the accomplishment of work at lower levels of the project schedule. Examples of detail milestones include release of engineering drawings, placement of a purchase orders for materials or sign-off of test procedures.

PCMs are key events within the project schedule which are considered critical. As such, they can be identified from any part of the logic network and can include major, contract or detail milestones. In addition to the example milestones listed above, PCMs might also include deliveries of flight hardware from industry suppliers, release of major builds of ground system software, successful completion of a prototype test, the release of a Request For Proposal (RFP) to industry, etc. They can also represent the completion of interim stages of work within a major activity. The major criterion for PCMs is simple but important: would missing the milestone threaten project cost, schedule or technical health? If the answer is yes, then it is a candidate for the PCM list.

PCM Illustrated

By way of illustration, Table 1 is the first page of the Project Control Milestone & Total Float Report for the hypothetical Meteoroid Identification & Space Tracking (MIST) Project under development by the TriStar Aerospace Corporation, which is the prime contractor for this NASA mission. The PCMs were identified from MIST's integrated project logic network. For example, the first PCM in Table 1 is MIST255 "Pre-Environmental Test Review" (PER). MIST255 is the activity identifier within the MIST schedule database which corresponds to the completion of the PER. Table 1 also contains the Baseline Delivery and Baseline Total Float columns, which refer to the delivery or completion dates and total float of the PCMs that were planned when the project schedule was baselined. Also included in Table 1 are the current (April) and prior (March) months' forecast delivery dates and total float. An actual PCM completion is identified with the letter "A" next to

'METEROID IDENTIFICATION & SPACE TRACKING (MIST) PROJECT
PROJECT CONTROL MILESTONE & TOTAL FLOAT REPORT

DATA DATE: 30APR96

ACTIVITY IDENTIFIER	ACTIVITY DESCRIPTION	BASELINE DELIVERY	BASELINE TOTAL FLOAT	MARCH DELIVERY	MARCH TOTAL FLOAT	APRIL DELIVERY	APRIL TOTAL FLOAT	TF CHANGE MAR / APR
MIST MILESTONES								
MIST255	Pre-Environmental Test Review (PER)	17MAY96	19	17MAY96	23	17MAY96	23	0
OBS242	Pre-Shipment Review (PSR)	17MAR97	15	26MAR97	9	02APR97	3	-6
OBS240	Observatory Ready for Shipment	27MAR97	11	05APR97	3	12APR97	-5	-8
OBS0248	Observatory Arrival at Launch Site	22APR97	11	01MAY97	1	08MAY97	-5	-6
OBS500	MIST Launch Readiness	01APR98	0	01APR98	0	06APR98	-5	-5
MIST250	MIST Mission Operations Review (MOR)	28MAR96	87	28MAR96	87	29MAR96(A)	0	0
POWER SUBSYSTEM								
POSA670	+Z Solar Array Panels Delivery	06MAR96	84	19APR96	52	10MAY96	44	-8
POSA695	+Z Solar Array Panels Ready for SADDS I&T	20MAR96	84	03MAY96	52	24MAY96	44	-8
POSA671	-Z Solar Array Panels Delivery	03MAY96	49	31MAY96	26	31MAY96	33	7
POSA696	-Z Solar Array Panels Ready for SADDS I&T	17MAY96	49	14JUN96	26	14JUN96	33	7
POBAT960	Super NiCd Battery Delivery	15APR96	152	30APR96	142	30APR96(A)	0	0
POBAT980	Super NiCd Battery Delivery (spare set)	13MAY96	152	29MAY96	142	29MAY96	144	2
C&DH SUBSYSTEM								
CDH6012	RTT A Ready for OBS I&T	22MAR96	49	12APR96	5	23APR96(A)	0	0
CDH6022	RTT B Ready for OBS I&T	28MAY96	5	28MAY96	5	04JUN96	8	3
ATTITUDE CONTROL SUBSYSTEM								
ACS402A	ACS B5.2 Ready for Formal S/W IV&V	15MAR96	35	14MAR96	0	14MAR96(A)	0	0
DEPLOYABLES SUBSYSTEM								
DES08021	+Z SADDS Flight Wing Ready for OBS I&T	04SEP96	12	12SEP96	2	03SEP96	14	12
DES08022	-Z SADDS Flight Wing Ready for OBS I&T	06SEP96	14	12SEP96	6	02OCT96	-3	-9
DES2016	SADA Ready for OBS I&T	15MAR96	10	18MAR96	0	18MAR96(A)	0	0

Table 1. Project Control Milestone and Total Float Report.

the date in the April delivery column. These ingredients comprise the fundamental elements of schedule reporting: baseline schedule, actual performance, current forecast and variance.

To describe this concept further, located under the subheading Power Subsystem, is the seventh milestone in Table 1: POSA670 "+Z Solar Array Panels Delivery." Again, POSA670 is the activity identifier which corresponds to the delivery to TriStar of the +Z Solar Array Panels from the Nova Corporation, the industry supplier. Upon delivery to TriStar the panels will be inspected and tested prior to turnover to the next higher assembly. As indicated in Table 1, the baseline delivery for the +Z Solar Array Panels was March 6, 1996 (early finish) with a total float of +84 days. In other words, if the +Z panel delivery is delayed beyond March 6, there are 84 days of float, or slack, available before this delay would impact the target completion date of the hypothetical MIST Project which is its launch date of April 1, 1998. Similar delivery and float status for the current month of April and the prior month of March are contained in the Project Control Milestone & Total Float Report in order to highlight variances against the baseline as well as the prior month's forecast. Float will be described in more detail under the section Control Milestone Analysis.

Lets examine why the POSA670 "+Z Solar Array Flight Panels Delivery" has been identified as a PCM in terms of the schedule, technical and cost health criteria described earlier. First, in terms of schedule health, a delay in the +Z Solar Array Panels could

mean a later than planned completion of the + Z Solar Array Flight Wing: the deployable subsystem of which the +Z Arrays are the critical component. Delays in Flight Wing build-up and test could further delay the MIST observatory integration and test program. Ultimately, the launch readiness could be in jeopardy.

Next, the technical health of the project could be threatened by a delay in this PCM. For example, further serious schedule delays with the +Z Solar Array Panels could result in a decision to eliminate or reduce the scope of downstream testing in order to meet the launch date. If the delay of this or any PCM resulted in a slip in the planned launch date, it could mean losing valuable science mission life and possibly lead to a significant cost overrun. In terms of cost, TriStar has a firm fixed price (FFP) contract with the Nova Corporation for the Solar Array Panels. With the exception of change orders, delays in delivery would not necessarily impact MIST's cost for the solar array panels themselves in terms of their development budget. While this direct cost may not be at risk in the case of further delays for this FFP delivery, there is almost certainly the additional indirect cost associated with: 1) the technical team's investigation into the problem 2) further project and procurement management attention, 3) additional travel funds to coordinate with Nova, 4) delay to the start of the next higher assembly, and 5) possible delay to the observatory integration and test program.

Therefore, delivery of the +Z Solar Array Panels from the Nova Corporation to TriStar is a critical milestone on the PCM list primarily for schedule reasons, although cost and technical elements are also considerations. As a first step, identifying the proper PCMs is an important part of providing the project manager with a concise set of the milestones that summarize the entire project schedule and provide a focal point for management control.

Establishing the Project Control Milestone Plan

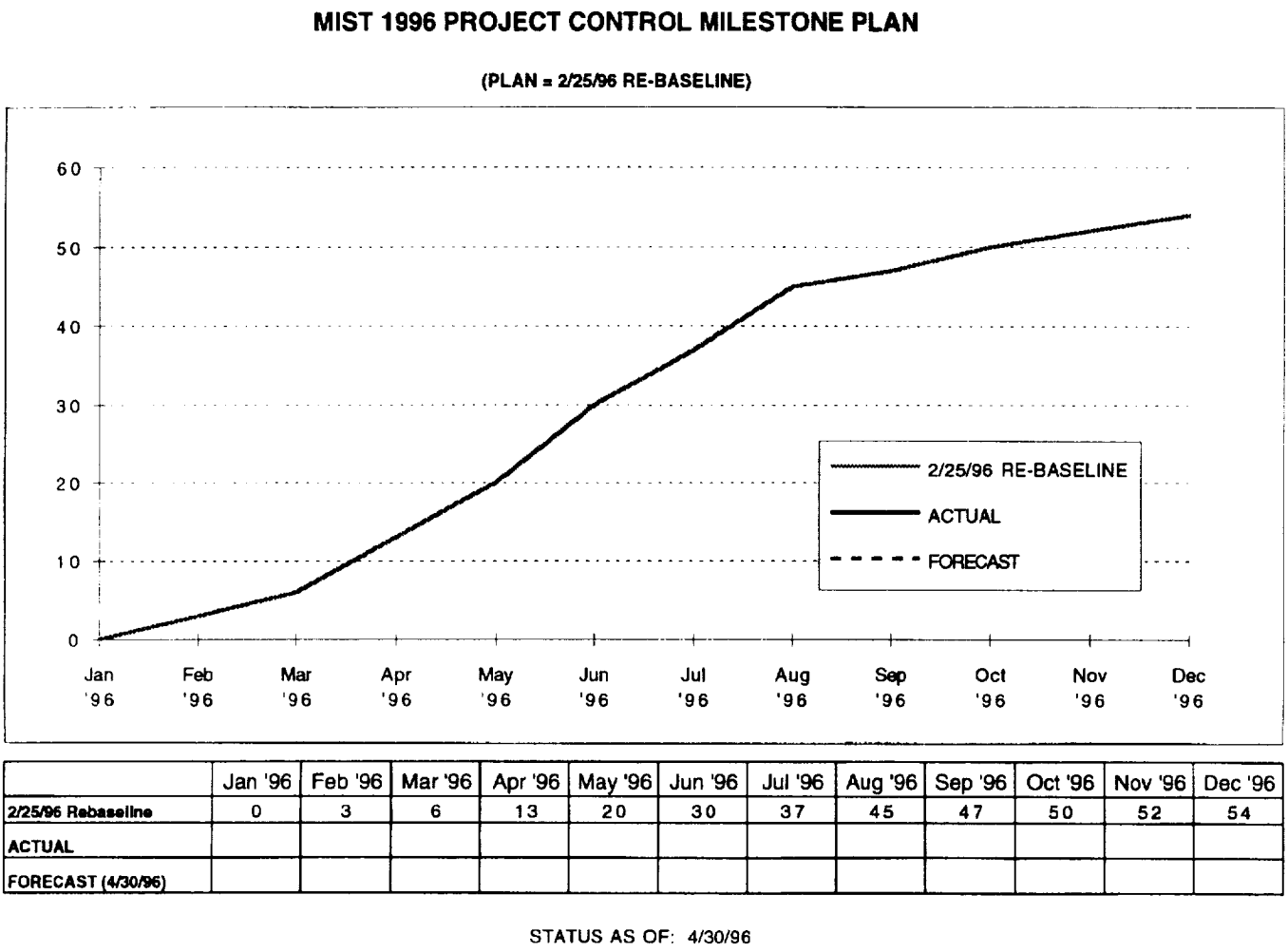
Once the PCMs have been identified, their corresponding planned completion dates (early finishes) can be easily depicted as a cumulative plan over time. MIST's cumulative PCM plan from its February 1996 rebaseline through December 1996 is summarized in Figure 1. Figure 1 was constructed simply by adding together each month's PCMs and plotting a cumulative curve. The cumulative curve is a logical format for depicting the PCM plan because its realism will be readily apparent in the conservative build-up, rapid acceleration and slow reduction in PCMs typical of the standard "S" curve. The same summary can be done for any period of time, depending on the needs of the project. For a project just getting underway, a summary of the PCMs leading up to the Critical Design Review (CDR) is a good starting point. Additional PCMs could be added in a "rolling wave" fashion as time elapses. The scale could be by week, month or quarter. This approach is similar to cumulative cost plans, drawing releases, etc.

It is important to emphasize that since the PCMs are drawn directly from the project logic network, the PCM plan is traceable to all levels of the project schedule: master, intermediate and detail. The PCM plan is not separate from, but part of, the overall project schedule. A PCM plan similar to Figure 1 conveniently summarizes what is expected to be accomplished over a fixed period of time.

With the PCM plan, the project manager now has a summary metric or way of measuring the schedule in terms of plan, performance and forecast-to-complete. This high level view of the schedule allows him or her to see the big picture, further enhancing schedule control.

Control Milestone Performance & Forecast

On a hypothetical major project such as MIST, the logic network is updated with the current status and forecast once each month to coincide with workforce and financial reporting. Since the PCMs are an integral part of the logic network, they are automatically updated each month when the network is statused. For example, in Table 1 the PCM ACS402A "ACS Build 5.2 Ready For Formal S/W IV&V"was actually completed on March 14, 1996. This actual completion date is identified by the "A" in the April delivery column. This means the build testing of atti-



	Jan '96	Feb '96	Mar '96	Apr '96	May '96	Jun '96	Jul '96	Aug '96	Sep '96	Oct '96	Nov '96	Dec '96
2/25/96 Rebaseline	0	3	6	13	20	30	37	45	47	50	52	54
ACTUAL												
FORECAST (4/30/96)												

STATUS AS OF: 4/30/96

Figure 1. PCM plan.

tude control subsystem software Build 5.2 was actually accomplished on March 14 and delivered to the IV&V laboratory for testing. The delivery of ACS402A allows credit to be taken for completing this PCM.

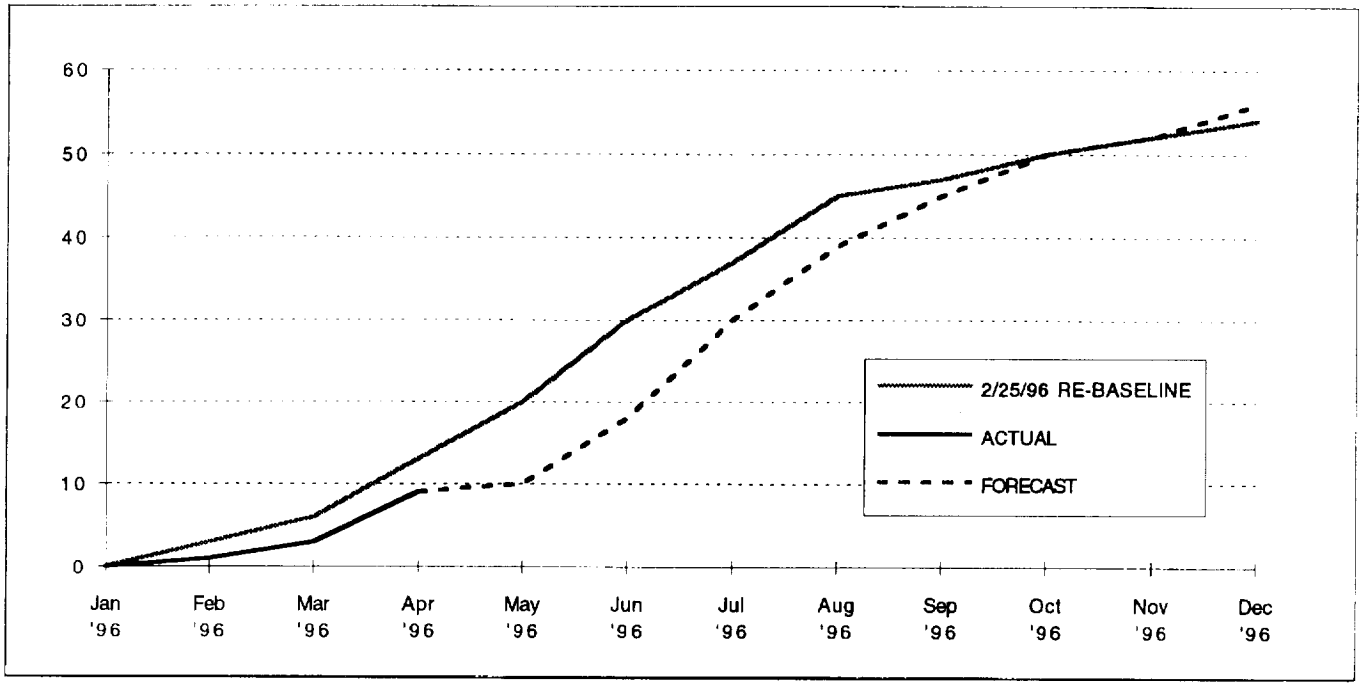
In addition to actual PCMs completed, the status cycle also provides the current forecast, or projection, of when remaining PCMs will be completed. Again, referring to Table 1, PCM CDH6022 "RTT B Ready For Obs I&T" has a baseline scheduled delivery of May 28, 1996, which was also last month's (March) forecast delivery. The current month's (April) forecast completion is June 4, 1996. This means that the Realtime Telemetry Tracker (RTT) B-side flight unit will be finished testing and delivered for integration with the MIST observatory on June 4, based on the forecast for completing the work remaining on it.

Once the schedule status accounting cycle is completed and the actual and forecast dates for the PCMs are obtained, PCM schedule performance is summarized by plotting the actual milestones completed and current forecast against the plan. Figure 2 illustrates the comparison of MIST's cumulative PCM actuals and current forecast to the PCM plan which was introduced in Figure 1.

Again, with a list of PCMs, the project manager can see at a glance what his or her project's major events are, when they are scheduled for completion and how much margin or float exists to accommodate possible delays. At the same time, the project man-

MIST 1996 PROJECT CONTROL MILESTONE PERFORMANCE

(PLAN = 2/25/96 RE-BASELINE)



	Jan '96	Feb '96	Mar '96	Apr '96	May '96	Jun '96	Jul '96	Aug '96	Sep '96	Oct '96	Nov '96	Dec '96
2/25/96 Rebaseline	0	3	6	13	20	30	37	45	47	50	52	54
ACTUAL	0	1	3	9								
FORECAST (4/30/96)					10	18	30	39	45	50	52	56

STATUS AS OF: 4/30/96

Figure 2. PCM performance metric.

ager has confidence in the realism underlying the plan and status because the PCMs are contained directly in the detailed project logic network.

Control Milestone Analysis

So far a basic approach to identifying PCMs and portraying their plan and corresponding performance has been described. This process should be taken a step further by analyzing what the performance data means and making an assessment of what to expect in the future for the project schedule. In the hypothetical MIST example illustrated in Figure 2, some important information can be obtained from the PCM performance metric. For the period ending April 30, 1996 (data or status date), 69% or 9 of the 13 planned PCMs were actually accomplished. The project manager can quickly gauge the overall schedule performance for the month as well as the cumulative performance to date and immediately focus on those major milestones that have not been accomplished. Variances to the plan are readily apparent, and specific PCM problems can now be investigated for cause and corrective action. Additionally, those PCMs that have not been completed in accordance with the baseline schedule indicate not only the amount of work still remaining, but suggest that performance efficiency may have to improve in order to get back on track.

For example, milestone POSA670 "+Z Solar Array Panels Delivery" was described earlier as one of the four PCMs not accomplished as of the reporting period ending April 30th. In Table 1 the project manager can see that its delivery has been delayed from the forecast April 19th delivery at +52 days total float

reported last month to the current forecast delivery of May 10th at +44 days total float reported in the current month, a reduction in float of eight days. In addition to a comparison of the current month's forecast delivery to last month's forecast, a comparison against the original baseline delivery of March 6th at +84 days float shows that the +Z Solar Array Panels are almost three months behind the baseline scheduled delivery and forty days of float have been consumed. Recall that total float is the amount of time an activity or event can be delayed before it impacts the project's completion point: the April 1, 1998, launch date in the case of MIST.

While it is a concern that this PCM has been delayed resulting in a loss of eight days of slack from the prior month, it is not yet a major problem. In this hypothetical example, a test equipment problem (cause) has been resolved by the technical team and a software patch (corrective action) has been incorporated by the Nova Corporation. Additionally, the remaining +44 days of schedule slack is still a sufficient margin should other unforeseen problems emerge. The value of the PCM reporting is that it alerts the project manager of significant schedule changes to critical project elements in order to facilitate investigation and implement corrective actions.

For a project that has implemented a performance measurement system (PMS), the PCM data provides a way to augment the variance analysis and schedule efficiency calculations. For example, the Budgeted Cost of Work Performed (BCWP or earned value) minus the Budget Cost of Work Scheduled (BCWS or the budget) indicates the Schedule Variance (SV), or difference between the dollar value of the work actually accomplished versus the work that should have been accomplished in the reporting period: SV = BCWP - BCWS. Similarly, the difference between the PCMs accomplished vs. planned could be compared to the formal SV. On a percentage basis the SV and PCM variance should correlate within a +/- 10% range. If not, then further investigation into the difference may be required.

Similarly, the Schedule Performance Index (SPI) = BCWP/BCWS. This ratio of work performed vs. work scheduled can be easily compared to the ratio of the number of PCMs accomplished vs. planned in order to gauge the relative efficiency of the schedule performance. The formal SPI and PCM ratio should also correlate within a +/- 10% range. If the SPI indicates 92% and the ratio of PCM actuals to plan is only 75%, it might suggest that the project schedule is not fully integrated with the PMS, earned value is being taken for work performed out of sequence, etc.

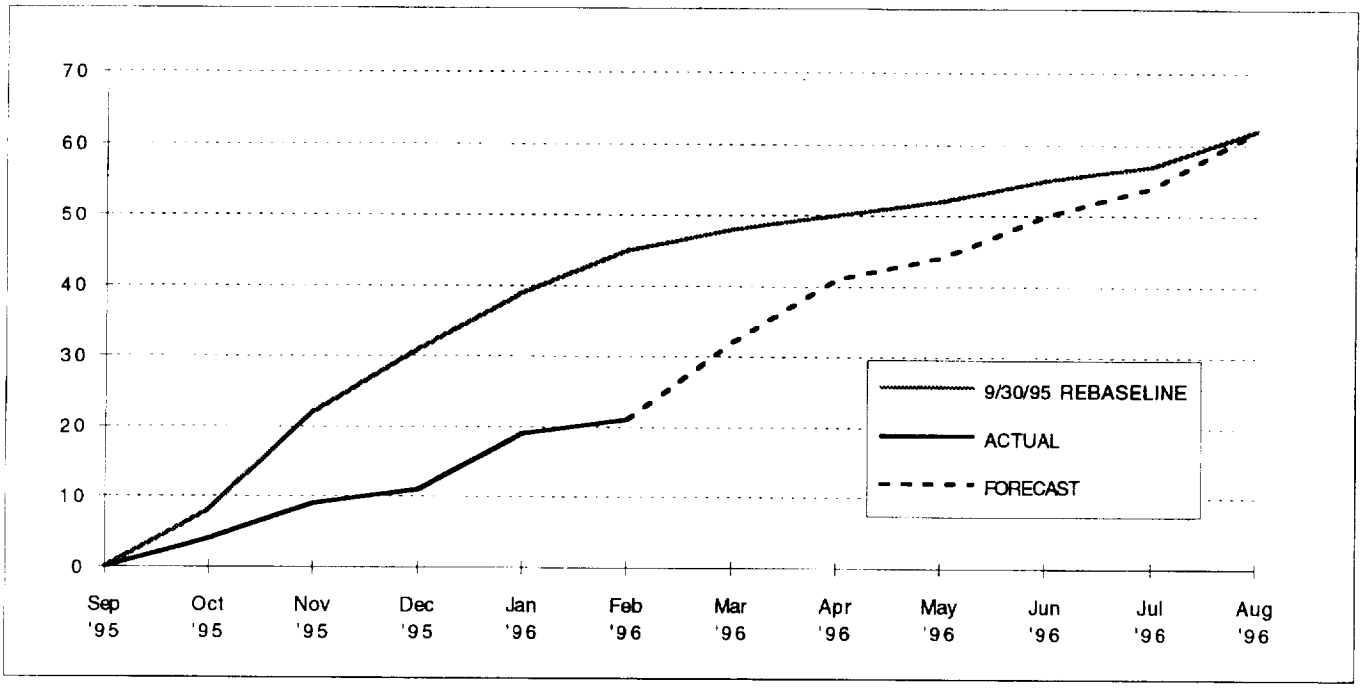
While the PCMs provide a measure of schedule performance, they also provide a good tool for trend analysis and insight into the realism of schedule forecasts, particularly when applied to the surveillance of contractor and supplier schedules. Consider Figure 3 which depicts the PCM plan, performance and forecast for the hypothetical Advanced Spectrum Analyzer (ASA). The ASA is a key scientific instrument for MIST being developed by the Browning Aircraft Company under a Cost Plus Award Fee (CPAF) contract from NASA. NASA, in turn, will provide the ASA as Government Furnished Equipment (GFE) to TriStar for integration into the MIST spacecraft. Figure 3 summarizes the PCM status for the ASA contract identified in Browning's logic network as of February 24, 1996. The NASA logic network is a Contract Data Requirement List (CDRL) item delivered each month to the MIST NASA Project Office.

Clearly, Figure 3 triggers a number of danger signals. First, note that the Browning is 53% behind the cumulative PCM plan through February 1996. Moreover, an alarming trend has emerged in that each month the actual number of milestones has fallen short of the plan. In fact, the Browning is averaging only 4.2 PCM completions each month. Also, another concern illustrated in Figure 3 is the classic case of the overly optimistic forecast. Note how the forecast, or estimate-to-complete, for the PCMs ultimately "catches up" in August 1996, while the performance trend suggests this is unlikely.

Although Figure 3 does not explain why Browning is not performing to plan or what the basis is for its optimistic schedule forecast, it does give the project manager a starting point for investigating the poor performance. Moreover, if caught early enough, proper management and technical attention can be

MIST ASA PROJECT CONTROL MILESTONE PERFORMANCE

(PLAN = 9/30/95 Nova Corp. Rebaseline/Estimate-To-Complete)



	Sep '95	Oct '95	Nov '95	Dec '95	Jan '96	Feb '96	Mar '96	Apr '96	May '96	Jun '96	Jul '96	Aug '96
ETC / REBASELINE	0	8	22	31	39	45	48	50	52	55	57	62
ACTUAL	0	4	9	11	19	21						
FORECAST (2/24/95)							32	41	44	50	54	62

SOURCE: ASA CDRL 005 3/20/96 STATUS AS OF: 2/24/96

Figure 3. PCM plan, performance and forecast.

applied to the underlying problems associated with such contracts. Otherwise, if left unchecked or without an improvement in efficiency, Browning's performance could continue to deteriorate, supported only by the claim that "things will get better next month." In fact, as described earlier, the ASA contract has been averaging 4.2 PCM completions per month since October 1995. A simple extrapolation of this rate suggests that the ASA will not complete all 62 of its PCMs until December 1996 if the present trend continues. This is four months after the planned delivery date of August 1996 (see Figure 3). This could result in potential technical and schedule problems for the MIST spacecraft integration program which needs the ASA instrument to continue into the test program. Moreover, severe cost overruns at the contractor could emerge if this condition continues. As a CPAF contract, the MIST NASA Project Office will have to allocate management reserve to cover the Browning overrun in order to complete the ASA instrument.

However, with careful surveillance of the supplier's schedule performance through PCM monitoring, the MIST Project would understand far in advance that the ASA instrument would probably be delivered much later than the Browning's estimate-to-complete indicated. In anticipation of the late ASA delivery, a workaround plan could be formulated to mitigate this problem. For example, the observatory integration and test sequence could be modified, resulting in a workaround plan that integrates the ASA before the start of the first observatory comprehensive performance test.

Whether for a total project or a key element of it—such as a major hardware item under contract with a supplier—a PCM approach to schedule control provides a framework for the project manager to understand the schedule status against the original baseline and the most recent replan. At the same time it affords a simple, graphical way of not only capturing trend data, but quantifying the amount of effort remaining to be done and the urgent issues which need attention.

A Project Control Milestone approach to monitoring schedule performance, forecasts and margins does not replace a conventional logic network schedule or other scheduling techniques. PCM metrics are simply a way to summarize a vast amount of schedule information for the project manager so he or she can understand the big picture and quickly assess potential schedule threats in order to take the appropriate corrective action. With the enormous number of technical, cost, procurement and administrative matters that demand the typical project manager's time, the PCM approach affords a way to quickly focus on the urgent needs of the project schedule and identify the elements that require immediate attention.

Systems Engineering: Three New Approaches

by Dr. Richard P. Evans

This paper describes three new systems engineering approaches: System Assessments, a Systems Integration (SI) program, and an Engineering Baseline System (EBS).

Some of the key features reported for System Assessments are an Assessment Control Board (ACB), as a critical complement to the traditional Configuration (or Change) Control Board (CCB), with associated one-page Assessment Plans (APs), and one-page Assessment Reports (ARs).

Primary characteristics of a Systems Integration (SI) program include the continuous acquisition of non-attribution System Reports (SRs); the structure of Candidate Program Initiatives (CPIs) for intermediate system planning; the application of small (3-5) consolidated customer/user/stakeholder and developer engineers in composite, non-attribution, altruistic Problem Area (PA) teams for system engineering review, and the use of Round Tables of participants in extracurricular roles, like INCOSE referees, to provide structured assessment support.

The Engineering Baseline System (EBS) addresses the opportunities/problems introduced by the recent widespread use of personal computers by engineers, the attendant separate and typically uncontrolled and non-standard structuring and naming of file-based system elements, and the accompanying associations, as new adjuncts to what had been exclusively a page-based environment. That uncontrolled and non-common creation and use of multiple separate file-based environments, and accompanying associations, brought on a loss of the standardized structure, naming, and change management that was previously maintained by the page-based environment—with its fixed and controlled page structure, page number, and page date.

The EBS paradigm includes standardized (thus common) system element structuring and naming (by a six-digit system number—that is a sequence number to sustain audits—and that has a six-digit suffix to support the assignment of unique system numbers to that span of separate system files. A system number has the format <<xxxxxx.yyyyyy>>. The xxxxxx prefix is a sequence number that is unique within the file or system component where the system element is maintained; and the yyyyyy suffix identifies that file. When a system element in a file changes, the next available system number prefix within that file is assigned; the suffix is fixed. All previous system numbers (prefixes) associated with a given system element are retained.

System numbers are unique for each system element, including specification elements, software code, drawings, and hardware elements. System numbers and associated tags, maintained in separate two-column ASCII-based index files, can be assigned by system developers when they create new system elements, without using specialized tools, or they can be assigned using database or CASE tool systems.

Added EBS features include the use of plain ASCII two-column index files for all manner of associations—even between code modules and user manuals—prepared, used and created by any and all engineers, anywhere, any time, for any reason—in contrast to the use of a central specialty database system.

That EBS element addresses shortcomings in the file-based approaches that are typically present in the current CASE-type environments. These approaches, while overcoming some of the page-based issues, but nevertheless based on the use of a few large, specialty tools, have also created problems and limitations of their own—particularly in the limited scope of those who are able to effectively participate.

Systems Engineering Principles

Three new methodologies are presented as systems engineering *approaches* in order to affect a shift from simply *engineering* to *methodologies for engineering.*

An example of one of the advantages of such a *transformation*—like a Laplace or Fourier transformation in addressing signal processing—is the following passage by Tully (1989) and Thome (1993), that is also illustrated in Figure 1, on the topic of systems engineering:

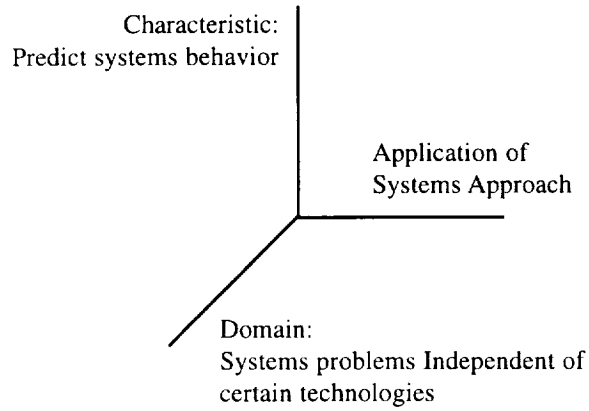


Figure 1. Systems Engineering—Three dimensions.

Systems Engineering consists of applying a systems approach to the engineering of systems. Its domain is the engineering of solutions to systems problems independent of employing a certain technology for realizing systems functions and properties. A characteristic of systems engineering is that it has to predict systems behavior and to design systems structure so that emergent behavior can be provided for and controlled within acceptable and desirable bounds.

In that approach, the authors address systems engineering along three separate dimensions that are more amenable to understanding and insight, as they transform in a sense from only *engineering* per se. That *transform* approach enables a separate consideration of each of the three dimensions, rather than addressing *engineering* as a whole. In the case of the dimension of *application of the systems approach*, for example, the transform effects a shift from the topic of engineering, to a separate consideration of the systems approach.

The authors, who also cite (Jenkins 1969 and Churchman 1989), then apply the same transform technique in considering the *systems approach* in the context of the following three primary *perspectives*, attributes of *systems thinking*, or *ways of thinking* about the engineering of computer-based systems, as depicted in Figure 2. The effective use of a three-dimensional framework for describing systems engineering and its various facets is similarly cited by Sage (1992), Hall (1969), and Warfield (1972).

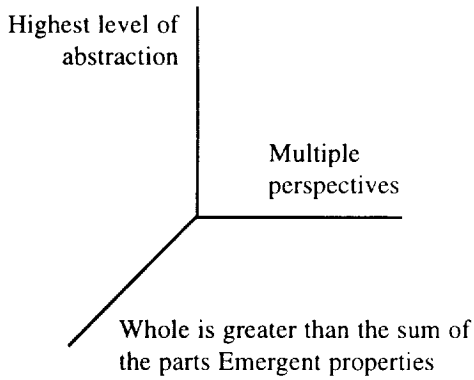


Figure 2. Systems Approach—Three dimensions (Perspectives, Ways of System Thinking)

The three new systems engineering approaches depicted in Figure 3, and presented in further detail in Sections 1, 2 and 3, respectively, are elements in this framework of systems engineering.

System Assessments and an ACB

Systems engineering approaches typically include a basic program control board known as the Configuration (or Change) Control Board (CCB). CCBs act after-the-fact in the sense that they receive formal change proposals in specific formats, some of which may have been in preparation for months. An Assessment Control Board (ACB) serves as a complementary and contrasting control board.

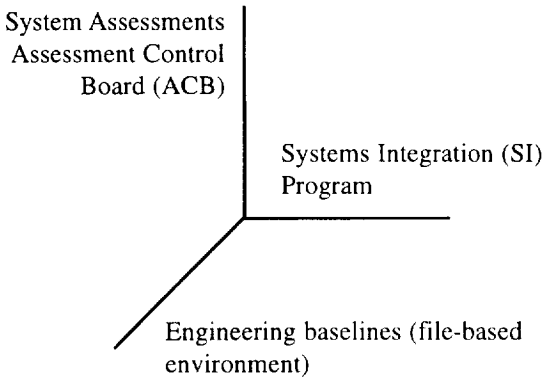


Figure 3. Three New Systems Engineering Approaches.

There is a need for both assessment control (ACB) and configuration control (CCB). Assessments make discoveries, a CCB disciplines the application of those discoveries. An ACB anticipates and plans, it operates up-front—that increases its leverage; a CCB operates after the fact and regulates. While CCBs are essential for change and implementation control, there is an equal need for the balance of assessment. An ACB, in contrast and as a complement to a CCB, is focused on the plans for the initiation of work, with a concentration on the plans for its assessment. In that sense, an ACB is focused on proposed plans and process, in contrast to a CCB emphasis on details of proposed change and the control of its implementation.

As illustrated in Figure 4, an ACB complements a CCB by exercising control of the initiation of work, including trade studies that lead to proposed changes for CCB consideration. The control of work initiation by an ACB includes the plans for, and the results of, the work assessment. An ACB focus is on assuring the operation of ACBs at all levels of the engineering effort, not just at the program office level. An ACB's goal is to assure a whole set of ACBs so that every engineer has the privilege to undertake their labors in the context of an Assessment Plan approved at an appropriate level by those to whom they also have the opportunity to provide reports of its application efficiently and effectively.

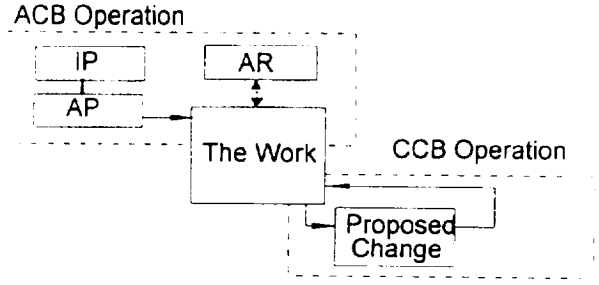


Figure 4. ACB and CCB Operations.

The initiation of work is controlled by an Initiation Plan (IP) approved by the ACB. An attachment to the IP is the one-page Assessment Plan (AP). Assessment results are similarly reported in typically one-page Assessment Reports (ARs). APs typically address the following:

Scope: The work and the associated products to be assessed.

Assessment Criteria: The criteria to be applied in assessing the work and the products. This is one of the hardest elements of a plan to devise, and accordingly one of the most critical program controls.

Approach: How will the assessment itself be assessed, how will the assessment be conducted—the format and process: who will be on the separate/independent assessment team—their names?

Schedule and cost: the assessment milestones and the proposed investment in assessment.

System Integration (SI) Program

A parallel methodology that can be applied to strengthen the CCB is for the ACB to also sponsor an SI program, as a complement to final CCB program control. The objective of an SI Program is to assure that proposed changes are well prepared for CCB consideration. Changes may be changes to the configuration of the program architecture and schedule, as well as a change to the design. There are three primary dimensions of an SI program as illustrated in

Figure 5, Identification, Investigation, and Implementation:

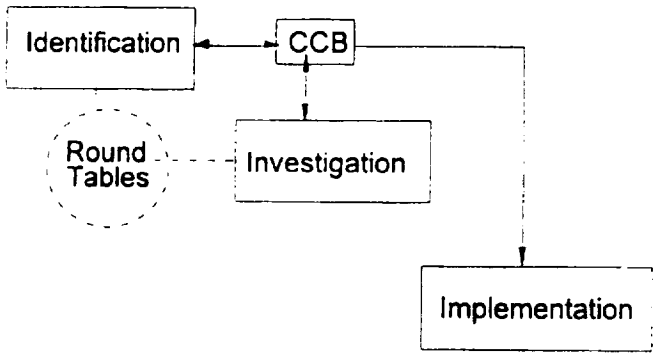


Figure 5. ACB Sponsored and CCB-Controlled SI Program

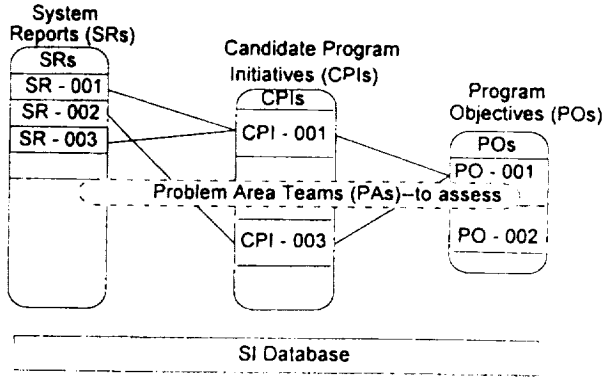


Figure 6. Four-part SI Program.

The driving influence of the SI Program is in the first two "I"s: Identification and Investigation—those that are the most *up-front*. The Investigation process also has, as a central feature, the use of Round Tables (RTs), as a panel of three to five experts, to serve like INCOSE referees as an unfunded assessment team for planned investigations.

The SI Program structure includes four elements: *System Reports (SRs), Candidate Program Initiatives (CPIs), Program Objectives (POs), and Problem Area (PA) Teams.* All are supported, as depicted in Figure 6, by an SI Database,

System Reports (SRs) are individually numbered records of every problem, suggestion, insight, or idea. An SI Database is built on the ever-accumulating set of SRs maintained throughout the life of the system. SRs are recorded as symptoms, so to speak, without prejudice. They are not filtered by any criteria, such as who said, or how they were reported, or whether they were validated. They are accumulated and honored by a unique SR Number that is never reused. Thus, while the SR may be placed in an inactive file, its identity, its number, always remains unique to that SR.

Problem Area (PA) teams assess the overall program handling of the SRs. The team members are drawn from both the user and the developer. They serve as professional collateral assignments, not as representatives of their parent organization's management priorities or interests. The PA teams assess; they do not have responsibility for solutions. They recommend initiatives, but they do not sponsor changes—with the attendant responsibility to implement approved changes. The PAs monitor the process design and operation.

Candidate Program Initiatives (CPIs) are temporary homes for potential program initiatives. CPIs are unfunded and do not have a designated management responsibility. They are the initial planning framework, a neutral territory, for the allocation of SRs. Note that SRs are allocated redundantly, with one primary allocation and multiple secondary assignments.

Program Objectives (POs) are funded, have assigned implementation responsibility, and are the formal vehicles for configuration change. POs are assembled as the implementation packages from the array of CPIs. They may be one entire CPI or include portions of many.

Engineering Baseline System (EBS)

The EBS methodology provides a new paradigm for system element identification, application, association, and control in the engineering of computer-based systems. Prior to the increasingly widespread use of computers by all engineers, system elements were only defined and controlled in a *page-based*

environment where the *page* structure, number, and date established system elements. With computers now available to, and in use by, essentially all engineers, a *file-based* environment is being added to the *page-based* foundation. The added *file-based* capability has both new promise as well as new risk; the EBS methodology addresses both. The EBS paradigm capitalizes on the file approach while addressing shortcomings in the typical CASE-type approaches to a *file-based* capability. Those systems, based on the use of a few large, central, specialty tools have, while overcoming some of the page-based-only issues, also created problems and limitations of their own. EBS features include:

1. *File-based engineering baselines:* prepared and controlled day-by-day in a distributed management framework.

2. *Standard common structure* of all system elements: controlled and defined at the basic primitive level as stand-alone, machine-processable elements. These are *file-based* structures that are structured *from* the *page-based* foundation.

3. Centrally assigned blocks of standard-format six-digit (auditable) *system numbers* that are maintained automatically in strict journal number sequence for every system element—whether requirements specifications, designs, test cases, maintenance documents, code modules, hardware components, budget elements, schedule milestones, or user manuals.

4. *Engineering baseline (eb) numbers, and engineering change (ec) numbers*, with associations to system numbers maintained in plain, two-column, ASCII index files for each primitive system element.

5. *Plain ASCII two-column index files* for all types of associations that are prepared, used and created by any and all engineers, anywhere, anytime, and for any purpose. These contrast to the use of a central specialty database system. EBS index files, prepared as individual two-column ASCII files, are thus not only amenable to being aggregated into larger sets of other plain ASCII files, they may also be aggregated into centralized, specialty, database-oriented software packages. Therefore, while not in any way constraining the use of specialty database-oriented tracing approaches, the index files actually enable them by enabling wide preparation and use outside of, and thus in support of, central database-oriented systems. On the other hand, using only specialty software applications, rather than *ASCII index files* to create as well as maintain associations, restricts visibility into those associations to either hard copy tables, or by direct use of the specialty software that created the table. Individual *index files*, however, remain visible to any and all for use, modification, extension, and review, and on any machine, and simultaneously also provide the needed inputs for a central database repository or report generator, as may be desired.

Problem Areas—Criteria for EBS Methodology Evaluation: The problem areas in current practice for the engineering of computer-based systems may be summarized in the following top-ten set of interrelated attributes. They are separate, but, as shown in Figure 7, they aggregate along three dimensions of system engineering needs (those that support, enable, and sustain all three dimensions are listed at the focus of the three axes):

- **Associations**—paired linkages of system elements.

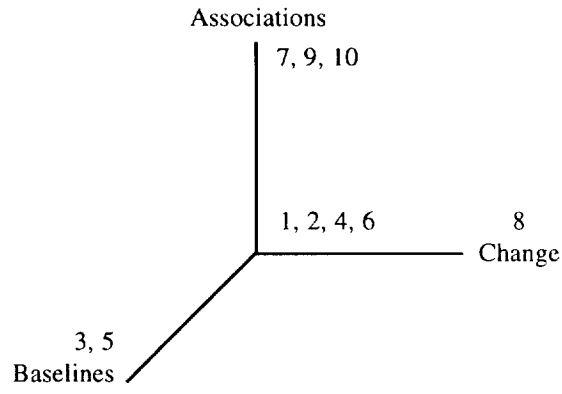


Figure 7. Three dimensions of systems engineering need as addressed by an EBS.

- **Change management support**, the identification and recording of changes, the associated rationale, and the specifics of new, changed and deleted system elements.

- **Engineering baselines**—multiple controlled baselines, maintained in distributed management environments, by, for and of the engineering.

1. Structure and granularity—common structure and system element number

2. Autonomy—stand-alone system elements

3. Timeliness—controlled engineering baselines as needed

4. Machine processability—ASCII files of systems elements and paired association

5. Distributed management—engineering groups with control of their own baselines—yet all integratable

6. Auditability—system numbers as sequence numbers

7. Self-rule—creation of paired association index files on the spot

8. Independence—non-dependence on hard copy only change definition

9. Aggregations—integration to one common database of separately controlled files—enabled by suffix block allocations

10. Associations—integration of all associations—ASCII paired index files

1. **Structure and Granularity:** The need is for controlled standardization of structure and naming/numbering to the lowest level; individual, uniquely numbered system elements that can also be separately processed in machines; CASE environments. Current ECBS controls are typically applied solely to formatted pages that are not machine processable without uncontrolled changes in structure. Current control practice also uses a framework of sections, such as 3.2.4.2.6, that often span sets of many otherwise separate requirements, specifications, and design elements. Further, current practice generally employs compound statements and bulleted and tabular data that are thus neither lowest-level system elements nor autonomous and stand-alone, as discussed below.

2. **Autonomy:** The ECBS methodology need is for stand-alone (as well as granular) system elements, that carry, with their unique name/number, all associated context and also the associated system/management information, including changes, allocations, associations/integration, and other system associations.

3. **Timeliness:** Effective engineering typically needs controlled *file-based* engineering baselines day-by-day. Current ECBS practice generally only provides formal *page-based* controlled baselines, and at *release* intervals that often span months, even years. Individual engineering activities usually need day-by-day controlled baselines for the interactions among their personnel, who are daily working on many tentative what-if type alternative assessments, designs, trade-offs, and other systems engineering considerations. They need day-to-day *engineering* baselines that are typically controlled among themselves. While those baselines are not the final contract type baselines that are eventually formally established by a CCB, equally formal control within their particular engineering activity is needed by them as they conduct their own iterative assessments and planning: the engineering.

4. **Machine processability:** The need is for system descriptions, whether specifications, designs, hardware components, software modules, etc., in ASCII non-formatted files—without dependence on features that are not machine-processable in ASCII files—such as tables, graphics, footnotes,

endnotes, italics, bold and indents. Tabular data is particularly susceptible to lack of machine processability as well as the attendant loss of automated auditability and change control. The same or similar data are often included in a variety of tables, with differing scope, format, and content. Thus change control, and even interface control, are difficult, if not precluded altogether. A controlled change to one table is not readily carried over to the needed changes in other tables as well as non-tabular system elements that address similar data, but in different formats and contexts.

5. **Distributed control:** Each engineering activity/organization needs to be enabled and responsible, to maintain a separate set of their own engineering baselines, yet integratable into a system whole. Current practice typically limits the authorization to establish baselines to a few centralized personnel using a large and unique specialty CASE tool or database.

6. **Auditability:** Names/numbers are needed that are centrally controlled, in a standard format (six digits) and strictly sequential—so that any missing or redundant number is clearly visible. Current ECBS practice relies on numbering/naming of system elements only by sections. They may include as many as 50 separate stand-alone system elements, with variable size numbers such as 3.4.2.1.3.7, and without separate, individual system numbers, of a fixed size number of characters such as 000357. In that framework, the only available names, for each system element is, for example, neither specific to each separate system element, nor is it a sequenced number to support audits. It is never assured, for example, that 3.1.6 would immediately follow 3.1.5.3.7.2; thus numbers may be missed. A sample of that inadequate *page-based* approach, along with its other association deficiencies, is presented in Table 1.

7. **Self-rule:** All engineers need to be both enabled as well as responsible to establish and maintain associations and dependencies—using the standard six-digit name/number—for all system elements they create and use. Current practice typically limits the establishment of associations to those entered by a few centralized personnel using a large and unique specialty CASE tool or database.

Document.	Function	Associated Segment
3.1.7.2	Provide on-line help	3.2.2.5 3.2.6.2.2

Table 1. Sample Page-based (Non-EBS-based) Traces.

8. **Independence from page-only change control:** Association of change data in each granular stand-alone name/number is needed. Current change management is typically based solely on change pages, without change definition embedded (by index files) with each separate stand-alone system element. Current controls are typically applied solely to formatted documentation that is not machine processable without uncontrolled changes in structure and associated change history.

As possibly one of the most significant benefits of the EBS paradigm for the engineering of computer-based system, change information is explicitly established and recorded for each system element, and that is maintained in individual machine-processable files and the associated two-column ASCII index files.

In the present practice on several large-scale systems currently in development, a major deficiency exists in the processing of formally approved changes, called RFCs, for Requests for Change. RFCs are allocated in composite sets to new *Versions* of formally controlled specifications, designs, budgets, schedules, test plans, installation manuals, etc. Each *Version* or *Release*, typically issued only after months of review by a CCB, normally includes several RFCs, with each RFC containing up to 10 pages, and with as many as 20 separate changes (system elements) per page. The RFCs are not structured to

primitive system elements for machine processing, and there is no unique identifier (like a system number) for each such basic change element. Further, there is no association index file (two-column paired associations between system numbers) for the individual changes in each RFC and each new revised system element in the composite *Version/Release*.

That deficiency is aggravated when the engineers remove the new information from the *pages* and enter them into machines. At that point, the non-association is compounded by the loss of the *page* date, number and structuring.

The following is an example of both the EBS approach to automatically recording (in two-column index files) changes to a given system element, and typical optional *display* formats. All system element information, including descriptions and index-file associations, is maintained in individual two-column index files. But various displays, such as the following sample, may be generated with various data aggregated on a page/report. In this case, para 3.2.1 was structured from the original in the *page-based* environment into two system elements in the *file-based* environment. Each was assigned a separate system number: 000002 and 000003, respectively. The second of those elements was altered by an engineering change (ec) action designated as <ec0827>. Please note that ec's may refer to formal RFCs or to any other controlled change process—especially those operated by the engineering staff as interim what-if changes. In the process, the engineering baseline (eb) increased from <eb0002> to <eb0003>. In addition, that new element was assigned the additional system number of <<000643.900001>>. Please note that the 2. <n2885> are for file ID (line number) "2", in the file named <n2885>.

2. <n2885> <eb0002> 3.2.1 The segment shall provide communications with the network through the Front End (FE) <<000002.900001>>.

2. <n2885> <eb0003> <ec0827> 3.2.1 The segment shall provide communications with the network through the Back End (BE) <<000003.900001>> <<000643.900001>>.

9. **Aggregations:** Use of system numbers with a six-digit suffix is needed to enable distributed baseline generation and control—yet integration (no conflicts in system numbers) into a single program database—aggregation of all management information into composite database sets of any needed scope. Allocation of "blocks" of suffixes (the "y") sustains this need: xxxxxx.yyyyyy

10. **Associations—integrations:** Each separate system element needs to be associated with all other related system elements by reference to its standard and unique six-digit system number/name—in paired ASCII index files—that engineers create without reference to any database.

Summary

System Assessment: An Assessment Control Board (ACB), with a focus on Initiation Plans (IPs), their associated one-page Assessment Plans (APs) and Assessment Reports (ARs), can be essential complements to CCB operations. CCBs are essentially totally after the fact. The resources (both time and money) to prepare proposed changes for CCB consideration have generally already been invested by the time the CCB receives the results. The operation of an ACB is *management working up front*, where the leverage is greatest. The use of IPs, APs and ARs at all organizational levels, operated in essence by increasingly lower-level ACBs, is a key feature of the ACB approach. The ACB influence of how work is to be assessed is a prime lever on what is done. The criteria for goodness and the names of those who will prepare assessment reports are key areas for management influence.

Systems Integration (SI) Program: Operation of an engineering planning process, as an SI Program, based on SRs as the primitives for all planning, is a potential added aid that the ACB can sponsor as a further complement to the CCB. An SI Program uses bipartisan Problem Area (PA) teams that include both customer and developer members. The PAs, working on a low duty cycle, an hour a week or less, concern themselves with the planning to address their assigned SRs.

Engineering Baseline System (EBS): Engineering baselines represent a new and needed paradigm for controlled visibility and traceability in systems engineering. The EBS addresses the opportunities and problems introduced by the recent advent of personal computers in use by all engineers and the attendant separate and typically uncontrolled and non-standard structuring and naming of *file-based* system elements and associated associations as new adjuncts to what was previously exclusively maintained as a *page-based* environment.

The EBS is more a methodological framework than a toolset. The EBS software is but one implementation of the approach. It exists to make real the principles; but the idea, the approach, and the concepts are essential. Not only is most of the so-called EBS conducted outside of any special software (engineers creating their own two-column index files in any type of tool—including paper and pencil), system development firms may quite readily construct their own software implementations of an EBS, once they appreciate and determine to employ the principles.

Acknowledgments: While there have been many who have assisted in the development of the methodologies described here, this article is the sole responsibility of the author, it does not reflect the views or opinions of any organizational affiliations.

References

Churchman, C. W. "Der Systemanatz und seine Feind," Translated from the American "The Systems Approach and its Enemies," commented and introduced by Werner Ulrich, Verlag Paul Hapt, Bern and Stuttgart, 1981.

Hall, Arthur D., "Three-Dimensional Morphology of Systems Engineering," IEEE Transactions on Systems, Science, and Cybernetics, Vol SSC-5, pp 156-160, Apr 1969.

Jenkins, G. M., "The Systems Approach," *Journal of Systems Engineering*, 1 (1969) 3-49.

Sage, Andrew P., *Systems Engineering*, John Wiley & Sons, New York, 1992.

Thome, Bernhard, (ed.), *Systems Engineering: Principles and Practices of Computer-based Systems Engineering*, John Wiley & Sons, New York, 1993.

Tully, C. J., "Position Statement on Systems Engineering," ATM/WP4.4/CJT7/Issue 1, 12 December 1989, Commercial in Confidence.

Warfield, John N., and Hill, Douglas J., *Unified Systems Engineering Concept*, Battelle Memorial Institute, Columbus, Ohio, 1972.

What If You Held A Best Practices Meeting—And Nobody Came?

by William R. Flury

"Hey! I've got a great idea! The Quality folks are always telling us that we should be on the lookout for better ways to do things. Let's hold a series of Best Practices workshops and see what ideas people bring us. We can email everyone to invite them and ask them to come prepared to talk about the Best Practices in their shops."

That's how it all began. We were discussing how we could get started with the business of process improvement and Mickey came up with the idea of holding a series of workshops where people could come together and discuss their Best Practices. We all thought it was a great idea. After we talked about it some more, we tried to figure out how many people might come and what we might have to do to prepare for the workshops. We agreed that we should test the idea by each of us calling some key people in our respective Centers and getting their reactions. Before we split we all agreed to make the calls and report back at our next meeting.

In order to keep the phone calls closely related to our topics of interest, we decided to focus them on the Key Process Areas (KPAs) of the Software Engineering Institute (SEI) Capability Maturity Model (CMM). We thought that we should limit the discussion to the Level 2 KPAs: Requirements Management, Planning, Project Tracking and Oversight, Subcontractor Management, Configuration Management, and Quality Assurance. Focusing on just a few practice areas should give us the biggest payoff. If we could identify the best practices in these areas we could endorse them as standards and start to spread them around.

What a surprise we got. The reaction to our calls was nothing like what we expected—but it did reveal a lot about our practices.

Reaction #1

"Gee, that sounds like a really great idea . . . but we don't have any Best Practices." These respondents said that they *do* all of the things that we talked about (i.e., the KPA items) but they always do them differently. They said that:

- Every job is different.
- Every customer is different.
- The staff comes from widely varied backgrounds and they learned different techniques in school, in other Centers, or that worked well on other projects.
- We just use what we think is best for each case.
- All of these get the work done, so it would be hard to choose which is best.

Reaction #2

"How would you ever decide which practices are *best*?" With all of the different types of tasks and all of the different procedures, methods, techniques, and tools in use, how would you ever begin to make some comparisons and evaluate differences?

What kinds of stories did we hear?

- Some things we do are in the textbooks . . . but we're not doing it exactly that way.
- If you asked five people how we do it, you'd get at least six answers.

- The methods keep changing.
- The staff keeps changing and the team does things the way they think will be best for the circumstances—and that varies by team experience.
- We don't really keep track from one project to the next on what we do differently.
- There might be some commonality among projects, but we don't keep any record of the techniques we use for any tasks. We rely on the memory of our key people.

We concluded that the most significant problem here was the fact that none of the practices was written down. There was no record of what practices were being applied to the different tasks and, as a result, it would not be possible to compare results of the use of different practices on similar tasks.

Reaction #3

"When you say Best Practices, who are they supposed to be *best* for?" Every set of practices requires a mix of resource inputs and provides a set of outputs. Everyone involved with those practices is affected in a certain way. Each person can determine a cost/benefit ratio associated with each practice. "So, who do we want to please with our Best Practices: (1) the customer; (2) our staff; (3) the supported and supporting systems with which our practices interface? For whom must we be best? That's a tough question."

"Looking at it another way, it's hard to figure at what we must be *best.*" Is the key (1) cost, (2) schedule, or (3) technical performance, or (4) some combination of the above? If it is to be a combination, what are the relative weights? Some people contend that it can all be reduced to a question of cost. Slipped schedules have an operational delay cost. Poor performance has a cost in rework—and, of course, cost overruns have a cost—but we don't have any good way to tally these.

We concluded from this that we would have to define our objectives better so that we could begin to do a better job of evaluation.

Reaction #4

"If we were to come to your meeting and describe how we do things—and others described how they do things, wouldn't a *Not Invented Here* (NIH) attitude prevail? How would we ever be able to convince others that our way is better than their way?" That's the reaction we're used to seeing. People come to meetings and talk about great ways to do various jobs and then they go back and continue doing exactly what they had been doing all along. Nobody ever comes forward with any convincing data—just opinions—and they don't sell.

Here's a summary of the situation.

- There are no standard practices—in fact there are not even any routine practices identified.
- There does not seem to be any basis for comparison among practices since we don't record which practices are used for various types of tasks and we don't record the outcomes.
- We don't seem to know how to determine *best.* We haven't decided what needs to be best and for whom.
- And, finally, we live in an engineering environment where we rely on facts to make our engineering judgments but, on the question of engineering practices, we have no facts, just opinions.

At the Next Meeting . . .

So, at the next meeting we decided it would be premature to try to hold the proposed workshops. We had to devise some way to start attacking the problems that had been raised.

We concluded that we should first focus on just one or two of the Key Process Areas and try to figure out how we should start to find our *Best Practices*. One thing was very clear: we had to get people to document their practices. We couldn't even begin to evaluate the practices until we could see them. So Step #1 "Get the practices documented."

Next, we addressed the question of what you would compare. In looking at Requirements Management practices, for example, we could look at:

- How many requirements were being handled,
- How clearly each was defined,
- How many TBDs were in the list,
- The stability of the list over time (amount of change),
- The verifiability of the requirements, and ultimately,
- The validity of the requirements.

We pondered how much recordkeeping might be required and concluded that it would not take much. We would need to record:

- The count of requirements;
- The number of changes per week or month . . . and the reason:
  - Lack of clarity,
  - Misunderstood customer,
  - Customer changed mind,
  - Tests could not verify meeting of requirement, and
  - Other; and
- The number of perceived shortcomings in the products after delivery (validity).

With those figures for any set of practices, we could begin to compare their relative performance with other sets of practices.

We looked also at planning. Maybe that would be even easier. For planning, we would just have to have good data on the planning estimates and the actual results. However, when we looked at this, it was just a bit more complicated. For one thing, we figured that we would need to know the basis of the planning estimates—how did we figure what effort each of the tasks would require? We would have to document that as a key element of the planning practice description. If "expert opinion" were being used, we would need to document the expertise. If an engineering roll-up were being used, we would need to document the work breakdown procedures that supported it. If a model were being used, we would have to know what model and the expertise of the operator using it. If standard rates were being used, we would need to know the source of the rates.

Our Action Plan Emerges

After hearing all the reactions and thinking them over, we decided to take some steps to start laying the foundation for identifying and evaluating the practices of our respective Centers. Here's what we decided:

1. We should encourage everyone to start writing down or drawing a picture of the steps in the current practices. This would be a necessary first step that would provide the foundation for all measurements and comparisons.

2. We should ask people to line up all the variations of each practice and see how they are the same or different. (We think that they will find much more commonality than they expect.) We will encourage the staff to agree on the common items and start to use them in the same way on all projects. We will also start to work with them to see how they can begin to evaluate the relative value of the variations so that they can decide which is best for their situation.

3. We will start people thinking about the concept of *Best Practices* supported by real data to prove their worth in various circumstances.

4. We will start publicizing the idea that everyone should adopt this approach as their first "best practice." We will begin by doing a newsletter article on the subject.

The above is a fictional account based on staff responses to mention of the possibility of holding some Best Practices workshops. The problems and the suggested solutions are real, the "meetings" are a literary device to keep the reader involved.

Software Reliability Assessment—Myth and Reality

by Myron Hecht, Dr. Herbert Hecht and Dr. Dong Tang

The importance of software as a contributor (if not the actual cause) of catastrophic events has been well documented (Leveson, 95). Moreover, as software is integrated into safety critical systems, the same quantitative reliability requirements which have been previously allocated to hardware are now being allocated to both hardware and software. For example, both U.S. Federal Aviation Regulations and International Joint Aviation Regulations impose maximum acceptable probabilities for failures of systems in passenger transport aircraft. Part 10 of the U.S. Code of Federal Regulations also establishes maximum acceptable probabilities for radioactive releases from nuclear power plants. When these standards were written, analog control systems were the dominant technology, and there was an accepted methodology for reliability prediction. Now digital (i.e., software-based) systems are replacing analog controls, but the old standards remain in force. The need for updating the standards and methodology extends to unregulated fields (e.g., computer-based automobile electronics), where there is economic motivation to being able to quantify the expected failure behavior.

The greatest need is for methodologies that can demonstrate that quantitative requirements are being met. More detailed quantitative characterizations are also needed to identify system bottlenecks and provide insight for decision making. An overview of the principal methodologies is presented in Table 1, and individual descriptions of each methodology follow.

Technique	Life Cycle Phase	Typical Measure	Advantages	Limitations	Predictive Power
Fault density	All (1)	Faults/KSLOC	Reference data available	Must assume encounter rate	Low
Reliability growth	Test	Failures/ execution hour	Some reference data available, objective measurement	Requires observation of multiple failures	Medium
Structured dependability	Test & operation	Failures/ execution hour for each segment	Models software structure, objective meas.	Few reference data, requires observations	Medium/high
Rare events	Operation	Failures/ operating year	Applicable to very high integrity systems	No reference data, requires observations	Potentially high

(1) Prior to the coding phase, a measure of deficiencies per estimated KSLOC can be employed.

Table 1. Comparison of Reliability Assessment Techniques.

Fault Density Model

The fundamental assumption behind fault density-based prediction models is that as the number of software coding defects (faults) increases, reliability decreases. The U.S. Air Force Rome Laboratory sponsored research into developing predictions of fault density (i.e., number of coding defects per thousand lines of source code) which they could then transform into reliability measures, such as failure rates (Friedman, 92). The predictions of fault density are based on the characteristics of the application, development environment, extent of reuse and other factors. This study and other sources contain data on expected fault density which currently ranges from 1 to 5 faults per thousand source lines of code (KSLOC). The translation of fault density to failure rate requires assumptions about the probability of encountering a fault during execution. This probability can vary widely, depending on the location and nature of the fault. The empirical data on this probability that are currently available do not support very accurate predictions of the failure rate.

Software Reliability Growth Models

Software reliability growth models use measured trends of failure rates (or change in intervals between failures) and extrapolate them to future operation. In most cases, they evaluate the reduction in failure frequency during successive developmental test intervals to estimate the software reliability at the conclusion of the test (and sometimes into operational deployment).

Reliability growth models have been an active area of research since the early 1970s (Farr, 93). Examples are the Schneidewind model, the generalized exponential model, the Musa/Okumoto Logarithmic Poisson model, and the Littlewood/ Verrall model (ANSI, 92).

Figure 1 shows an example of such a model. The software is executed over a certain time interval, represented as T_n, until a failure occurs. The time between failures defines a hazard rate. It is expected (but not required in this particular model) that overall, the hazard rate will decrease over time, but that there are discontinuities as each failure occurs. However, as the program runs for more time, there is increasing confidence in the reliability of the program. Applications of these models have all been demonstrated using real data from software with typical failure rates of 10^{-1} to 10^{-3} per hour (Abdel-Ghaly, 86, Musa, 87).

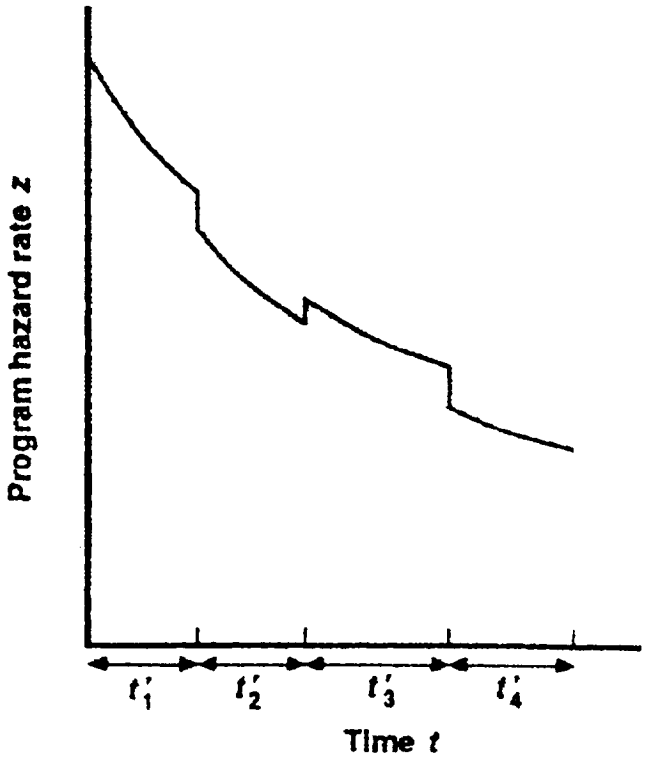


Figure 1. Reliability growth model.

Because of the very low failure rate required for life-critical software, reliability growth models and traditional testing techniques are not suitable (Butler, 93). For example, it would take 108 to 1010 hours (thousands of years) of testing to demonstrate a failure rate of 10^{-7} to 10^{-9} per hour, assuming one copy of software would be tested and one failure would be observed (Butler, 93). Even if 10 copies of the software are tested concurrently, it would still take hundreds of years. The study also cited comments of other experts in the field on this issue, including the following:

Clearly, the reliability growth techniques are useless in the face of such ultra-high requirements. It is easy to see that, even in the unlikely event that the system

had achieved such a reliability, we could not assure ourselves of that achievement in an acceptable time. (Littlewood, 93)

Another limitation of reliability growth models is their lack of ability to model software structure. Reliability growth models treat the software as a black box and form a single expression for its reliability. Critical software systems include fault tolerance mechanisms, such as error detection and handling, redundancy management and back-up tasks. As such, the reliability of the whole software system cannot be simply quantified by the number of failures observed at the component level. For example, a transient task failure may be covered by the fault tolerance provisions and may not affect critical functions. This scenario has been verified by several studies (Lee, 93; Tang, 95) which showed that 80 to 95 percent of software failures in real time fault-tolerant systems are recoverable by redundant processes. In such a case, reliability growth models do not provide meaningful answers, and structured dependability models must be used.

Structured Dependability Models

An alternative approach uses structured measurements, similar to the established hardware practice. In this technique, application software tasks, operating system kernels or executives, and hardware components are regarded as equivalent elements in a system. The operating times, failure rates, correlated failure probability, recovery times and recovery probabilities of any of these elements can be measured in reliability tests.

Reliability and availability are then estimated by models of the system structure, using measurement-based parameters for each component (Tang, 95). Statistical estimation of reliability and availability parameters and reliability modeling based on these parameters has been a research topic in computer engineering for 15 years (Iyer, 93). These analyses are based on operational logs and failure data.

Dependability models have been used to evaluate operational software based on failure data collected from commercial computer operating systems for about ten years (Hsueh, 87; Tang, 92; Lee, 93). The methodology has been extended to evaluate availability for air traffic control software systems in the late testing phase (Tang, 95) and most recently to the early operational phase at multiple sites.

In our experience, three model structures have been found useful in measurement-based dependability evaluation: the reliability block diagram, the k-out-of-n model, and the Markov chain. Both reliability diagrams and k-out-of-n models are combinatorial models and typically assume failure independence among modeled components. Markov chains are stochastic models that can incorporate interactions among components and failure dependence in the model.

However, the current practice of measurement-based evaluation for individual software systems (with the number of installations <100) is still limited to failure rates of 10^{-2} to 10^{-5} per hour and an availability of three to five 9's (0.999 to 0.99999). For example, the newly developed FAA Voice Switching and Control System (VSCS) is being installed in 21 major U.S. air traffic control centers. The system availability (dominated by software) was evaluated to have five 9's as of March 31, 1996. If no major failure occurs in the future, it would take 15 years of normal operation of the 21 systems to demonstrate an availability of the required seven 9's at the 80% confidence level.

In the process of collecting and analyzing such data, additional studies can be undertaken for more detailed examinations of underlying causes. For example, analyses of workload and failure data collected from IBM mainframes (Butner, 80) and DEC minicomputers (Castillo, 81) revealed that the average system failure rate is strongly correlated with the average workload on the system. Recent studies of data from DEC (Tang, 92) and Tandem (Lee, 93) systems showed that correlated failures across processors are significant in multicomputers, and their impact on dependability is significant.

The underlying assumption in these measurement-based approaches is that the fundamental failure mechanisms are triggered stochastically, i.e., are

non-deterministic ("Heisenbugs"). However, there is a class of failures in which the software runs to completion but produces an unacceptable output. For example, an electronic speed control on a turbine may in fact not shut down the device in an overspeed condition even though there was no crash, hang, stop or delay failure. This deterministic failure condition may be traced to a logic fault in the code or an incorrect set of parameters (e.g., the RPM threshold for that particular turbine under the specified set of pressures and temperatures). However, the root cause of the failure may in fact lie much deeper, i.e., defects in the system requirements or software requirements.

The techniques and methodologies for estimating the probabilities for these deterministic incorrect response failures are very immature. It is tempting to "wish them away" by positing that an adequate V&V (verification and validation) or integration testing program should uncover them. However, resources are finite, and it is rarely feasible to provide sufficient time or money to perform the level of testing needed to uncover all such failures, even in systems designed for high dependability. From a practical perspective, when estimating software failure rates, one should look not only at failures that cause losses or delays of system services (e.g., crash, hang, stop) but also incorrect response failures. If there are incorrect responses at the final stages of testing or integration, or in initial operation, then reliability predictions made exclusively on the basis of stochastic failures may not be valid.

Obtaining adequate data from which to assess reliability and availability is critical to any measurement-based methodology. This obvious principle can be difficult to implement in practice for dependability assessments because of the constraints of an expensive testing program or impending project deadlines. Adequate data means monitoring and recording events of interest such as failures and recoveries of components, as well as performance parameters of the target system while it is operating under representative workloads. It also means collecting data on failure modes so that an assessment of the importance of deterministic failures can be made. The events and parameters to be collected should be representative of the system operation and meaningful for the assessment of the system. Measurements should be made continuously for a sufficient period to yield statistically significant data. Operating logs should include information about the location, time and type of the error, the system state at the time of failure or abnormal operation, and error recovery (e.g., retry) information where applicable.

Assessment by Rare Events Technique

As previously discussed, none of the techniques described above can furnish a credible direct assessment for failure rates lower than 10^{-6} per hour. Under favorable circumstances, the structured dependability approach may support the conclusion that such requirements are met by two or more independent versions running under a highly reliable selection or voting scheme, and this is indeed the way adopted by many exacting applications. It is an expensive solution, because in addition to the multiple software implementations it requires the development and very extensive testing of selection mechanisms. Further, multi-version software tends to degrade the computational performance (because of the need to wait for the slowest version to complete execution and related issues), and the independence of the versions cannot be taken for granted (because they implement a common set of requirements). Therefore, there is ample motivation to investigate other assessment techniques.

The basic premise of the rare events approach is that well-tested software does not fail under routine input conditions, which means that failures must be triggered by unusual input data or computer states. This assumption is validated by a number of investigations that are summarized elsewhere (Hecht, 94). Late-phase testing will usually subject the program to test cases that emphasize these rare conditions, and this permits assessment of the failure probability by the likelihood of encountering the rare conditions that triggered the failure rather than by test time.

As an example, consider a program that failed twice during the last 1,000 hours of test. The first failure occurred on restart after a simulated power interrup-

tion, while at the same time one of the input signals faulted to zero (sensor fault). The second failure occurred when one out of three inputs faulted to high and another one to low. Is the failure rate of this program 2 X 10^{-3} per hour as computed from the test time?

Most observers would disagree with such an assessment and will find it more reasonable to take into account the occurrence rate of the triggering events in the environment in which this program will operate. Assume that power interruptions normally occur only once a year, and sensor failures to zero are expected to occur only once every two years. The combined probability of the joint event (assuming the individual triggers to be independent), is therefore well over 10^{-7} per hour. The second test case that triggered a failure (one sensor high and one low), has an even lower probability. After the software has been modified so that it will not fail again due to these triggers, its failure probability will be much lower than that computed from the test time.

A quantitative assessment will consider the total number of test cases that had been used and the probability of the natural occurrence of the simulated conditions. To illustrate the basics of the quantitative assessment, assume that during the 1,000 hours of test there were 10,000 test cases that simulated conditions that are expected to arise more frequently than once per 10,000,000 hours and 1,000 test cases simulating conditions that are expected to occur less frequently. Since the only failures observed were due to the second category, and since there was a ten-fold greater opportunity for failures under the first category, it can be reasoned that the failure rate in the natural environment is expected to be not more than 10^{-7} per hour. The mathematical formulation of this approach is based on the probability of drawing black and white balls from an urn (Hecht, 96).

Conclusions

Reliability assessments based on fault density and reliability growth models support planning and comparative evaluations but are usually not sufficiently validated to be a credible basis for stating that a software product has attained a required reliability, particularly when the required reliability is high. Structured dependability models can furnish estimates that are more precise and that also identify the elements where reliability improvement will provide the greatest benefit. They are well suited for designing and maintaining highly dependable computer systems intended for flight control, ground transportation, air traffic control and nuclear power plant safety functions.

Except under unusually favorable circumstances, none of these methods can currently assess whether a software product meets requirements for failure rates of less than 10^{-6} per hour. The rare events approach, described in the preceding section, has the potential for being useful for applications that demand the highest dependability, but it is the least validated of the methodologies discussed here. Because of the constantly increasing use of software-based systems in critical applications, further research into software reliability assessment is urgently needed.

References

A. A. Abdel-Ghaly, P. Y. Chan and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," *IEEE Transactions on Software Engineering*, vol SE-12 no. 9, September 1986, pp. 950-967.

"American National Standard, Recommended Practice for Software Reliability," American National Standards Institute, ANSI/AIAA R-013-1992.

R. W. Butler and G. B. Finelli, "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software," *IEEE Transactions on Software Engineering*, vol SE19 no. 1, pp. 3 - 12, January 1993.

S.E. Butner and R.K. Iyer, "A Statistical Study of Reliability and System Load at SLAC," Proc. 10th Int. Symp. Fault-Tolerant Computing, pp. 207-209, Oct. 1980.

X. Castillo and D.P. Siewiorek, "Workload, Performance, and Reliability of Digital Computer Systems," *Proc. 11th Int. Symp. Fault-Tolerant Computing*, pp. 84-89, July 1981.

W. H. Farr and O. Smith, *"Statistical Modeling and Estimation Functions for Software (SMERFS) - User's Guide,"* NSWCDD TR84-371, Revision 3, September 1993.

Michael Friedman, "Methodology for Software Reliability Prediction and Assessment" Report RL-TR-92-52, Rome Laboratory 1992 (2 volumes).

Herbert Hecht and Patrick Crane, "Rare Conditions and their Effect on Software Failures," *Proceedings of the 1994 Reliability and Maintainability Symposium*, pp. 334 - 337, January 1994.

Herbert Hecht and Myron Hecht, "Quality Assurance and Testing for Safety Systems," *Proc. CADT-ED*, Beijing, July 1996.

M. C. Hsueh and R. K. Iyer, "A Measurement-Based Model of Software Reliability in a Production Environment," *Proceedings of the 11th Annual Computer Software and Applications Conference*, pp. 354-360, October 1987.

R.K. Iyer and D. Tang, "Experimental Analysis of Computer System Dependability," Technical Report CRHC-93-15, Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign, July 1993.

I. Lee, D. Tang, R.K. Iyer, and M.C. Hsueh, "Measurement-Based Evaluation of Operating System Fault Tolerance," *IIEEE Transactions on Reliability*, pp. 238-249, June 1993.

Nancy G. Leveson, *Safeware*, Addison Wesley, Reading, Mass., 1995.

D. Tang and R.K. Iyer, "Analysis and Modeling of Correlated Failures in Multicomputer Systems," *IIEEE Trans. Computers* Vol. 41, No. 5, pp. 567-577, May 1992.

D. Tang and M. Hecht, "Evaluation of Software Dependability Based on Stability Test Data" *Proc. 11th Int. Symp. Fault-Tolerant Computing*, Pasadena, California, June 1995.

Trends in Systems Engineering Life Cycles

by Dr. F. G. Patterson Jr.

The essence of life cycle management is division of resources (6). Kingsley Davis recognizes division of labor as one of the fundamental characteristics of socialized man (9). The division of tasks into subtasks, the documentation of progress through intermediate products: these are important concepts that did not begin in abstraction, but in concrete problems. The essence of engineering is problem solving; and divide-and-conquer strategies, process definition and improvement, process modeling, and life cycle management are all engineering methods that begin by reducing complex challenges into tractable parts and conclude by integrating the results.

The failures of some systems engineering efforts have led some theoreticians to criticize or abandon the traditional ways of dividing resources. The trend is broad and can be seen in many areas. Life cycles are criticized for many reasons. For example, the concept that a requirements phase is self-contained, self-supportive, and separable from other phases has come under sharp attack both from academia and industry. The deeply ingrained tradition in industry that requirements must not dictate any given specific design is also coming into question. The justification for this belief may be readily derived from our basic activity model, shown in Figure 1.

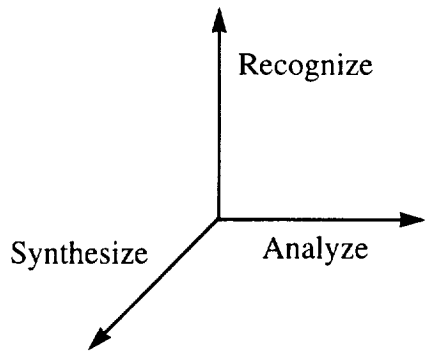


Figure 1. Engineering activity model.

Each of the three orthogonal activities shown in Figure 1 is necessary for successful systems engineering. It is not clear, however, that separation in time, separation by assignment to more than one action team, or separation in terms of any other resource based on the orthogonality of a process model (that is to say, a *life cycle*) is effective in reducing the engineering complexity problem. In fact, the opposite seems to be true. In the natural course of raising and resolving problems, it is much more efficient to resolve problems when and where they arise. The inefficiencies associated with following each life cycle step to completion before beginning the next step tend to result in a loss of information, and this tendency is exacerbated in proportion not only to the size of the engineering problem but also to the size of the engineering resources (5). This is a major obstacle in engineering big systems.

A solution based upon partitioning big systems into a number of small systems reduces the inefficiency inherent in life cycle-based approaches. However, this solution does not answer the emerging generation of systems engineering theorists who maintain that life cycle activities are dependent upon each other in a non-trivial way that neither a waterfall model, nor even a spiral model in its full generality, can adequately address. This suggests that specialties, such as *requirements engineering*, are undesirable, unless their scope of activity spans the entire life cycle: definition, development, and deployment. An alternate way to view this suggestion is that the number of specialties is effectively reduced to one: *systems engineering*. The systems engineer is ever-cognizant that a system must be specified, built, tested and deployed in terms not only of its internal characteristics, but also in terms of its *environment*. Thus, the job of integrating becomes a global concern that subsumes all of the steps in the life cycle.

Process Improvement

The trend away from the use of prescribed standard life cycles, while widespread in the private sector, has only recently begun to affect government acquisitions. The U.S. Department of Defense no longer requires the use of MIL-STD-499a for the acquisition of systems. Moreover, the new software development standard, MIL-STD-498 (10), after years of development, has been canceled without replacement. Other military standards are receiving similar treatment. The idea is that developers have their own processes that are specific to their own organizations. This change in emphasis should not be viewed as freedom from the use of a disciplined approach, but rather freedom to customize the approach to optimize quality attributes based upon variable factors in the software development organization.

The wide acceptance of ISO 9000 as an international standard is actually a trend away from standardized process models. The basic philosophy of ISO 9000 has been summarized as: "Say what you do; then do what you say" (14,23). ISO 9000 certification (15) is a goal to which many companies aspire in order to gain competitive advantage. Certification demonstrates to potential customers the capability of a vendor to control the processes that determine the acceptability of the product or service being marketed.

The Software Engineering Institute has developed a method of process assessment and improvement in the software development arena, known as the Capability Maturity Model (13,22,27). As the name suggests, the model is based upon the existence of a documented and dependable process that an organization can use with predictable results to develop software products. In effect, the details of the process are of little interest, as long as the process is repeatable.

The CMM model was adapted from the five-level model of Crosby (8) to software development by Humphrey. The five levels of the CMM model are:

1. The initial level: *ad hoc* methods may achieve success through heroic efforts; little quality management, no discernible process; nothing is repeatable except, perhaps, the intensity of heroic efforts; results are unpredictable;

2. The repeatable level: successes may be repeated for similar applications. Thus, a repeatable process is discovered which is measurable against prior efforts;

3. The defined level: claims to have understood, measured and specified a repeatable process with predictable cost and schedule characteristics;

4. The managed maturity level: comprehensive process measurements enable interactive risk management;

5. The optimization level: continuous process improvement for lasting quality. According to Sage (25), "There is much double loop learning, and this further supports this highest level of process maturity. Risk management is highly proactive, and there is interactive and reactive controls and measurements."

The CMM helps software project management to select appropriate strategies for process improvement by examination and assessment of its level of maturity, according to a set of criteria; diagnosis of problems in the organization's process; and prescription of approaches to cure the problem by continuous improvement.

Even though managers may be seasoned veterans, fully knowledgeable about the problems and pitfalls in the engineering process, they may disagree with each other on how to cope with problems as they occur. If agreement is difficult to produce in an organization, the resultant lack of focus is taxing on organizational resources and may endanger the product. Thus the management of an organization must be greater than the sum of its managers by providing strategies for management to follow and tools for management to utilize. Such strategies and tools will be the result of previous organizational successes and incremental improvements over time, and measured by a level of maturity. The Capability Maturity

Model (CMM), developed at the Software Engineering Institute (SEI) at Carnegie Mellon University, provides a framework that is partitioned by such levels of maturity. Although the CMM was developed to measure the maturity of software development processes, the ideas upon which it is based are quite general, applying well to systems engineering and to such processes as software acquisition management, and even the software engineer's own personal software process (12).

In an analysis of the CMM, it is helpful to look closely at the five levels or stages in quality maturity attributable to Crosby (8) in order to see how one level builds upon another. They are:

1. Uncertainty: confusion, lack of commitment. "Management has no knowledge of quality at the strategic process level and, at best, views operational level quality control inspections of finished products as the only way to achieve quality."

2. Awakening: management wakes up and realizes that quality is missing. "Statistical quality control teams will conduct inspections whenever problems develop."

3. Enlightenment: management decides to utilize a formal quality improvement process. "The cost of quality is first identified at this stage of development which is the beginning of operational level quality assurance."

4. Wisdom: management has a systematized understanding of quality costs. "Quality related issues are generally handled satisfactorily in what is emerging as strategic and process oriented quality assurance and management."

5. Certainty: management knows why it has no problems with quality.

In each of these environments, a particular kind of person is required. There is a shift in focus from one type of key individual to another as we move from one CMM level to the next. The progression seems to be, roughly, as follows:

1. **Heroes.** Necessary for success in a relatively unstructured process, the hero is able to rise above the chaos and complete a product.

2. **Artists.** Building on the brilliance of the heroes, the artists begin to bring order, resulting through repetition in a codifiable process.

3. **Craftsmen.** These are the people who follow the process, learning from experience handed down from previous successes.

4. **Master craftsmen.** These are the people who are experts in their respective facets of the development process, who understand and appreciate nuances of process and their relationship to quality.

5. **Research scientists.** Finally, master craftsmen appear, who, through experiential learning and attention to process integration, are able to fine tune the process, improving the overall process by changing steps in the process while avoiding harmful side effects.

The characteristics of the organizational culture are directly related to organizational learning. The organization appears to depend primarily upon two factors: (1) the people who compose the organization, and (2) the environment internal to the organization. Of course, a case may be made for including the external environment, since the overall success of the organization (and its probability of survival) are directly related to its adaptation to both the market and technological factors. Some of the organizational characteristics may be organized in five stages, following the CMM model, as follows:

1. **Heroes and supporters.** Dependent upon the ability of heroes to rise above the chaos, the organization grows up around the activities of each hero, each of whom may require low-level support services. The hero's processes are largely self-contained, and very loosely coupled with other heroes' processes. While there are very efficient aspects of this kind of organization (viz., the hero's own activities), there is no overall efficiency induced by integration of activities

into an overall process. Thus, at this CMM level, there are really two levels of workers: heroes and others.

2. **Artist colony.** Through mutual respect and attention to the successful practices of the previous generation of heroes, these people work together to recreate the successes of the past, creating new processes along the way. Management begins to be able to track progress.

3. **Professional cooperative organization.** Through long service and attention to process, master craftsmen have emerged, creating more hierarchical structure in the organization as less experienced individuals are able to learn from more experienced craftsmen. There now exists the concept of "the way to do the job," a concept that must be adhered to measurably. Management's role is to control adherence to the process by defining metrics and implementing a metrics program.

4. **Society of professionals.** At this point, the organization is mature enough to be able to receive from its individual members meaningful suggestions on how to improve selected parts of its process and to implement them in the overall process. This is largely a shift in the organization's ability to learn, of becoming a "learning organization."

5. **Institute of professionals.** The organization is now so mature that it is able to look continuously for ways to improve processes. Outside influences are no longer repelled, but are welcomed and evaluated.

In parallel with the trend to decommission development standards, there is a trend to buy off-the-shelf products instead of customized or in-house developed systems. When products may be found in the marketplace that meet the requirements of customers, economy of scale in manufacturing may lead to substantial cost savings over the cost of custom development. Even when products are not available in the market, the trend among large customers, such as the U.S. Department of Defense and the National Aeronautics and Space Administration, is to transfer the risk of in-house development from the customer to the contractor. Performance-based contracting is a tool that allows a customer to issue product specifications and acceptance criteria, and a supplier to collect a fee for creating the product as specified. Two significant differences between performance-based contracting and conventional contracting methods are:

1. There is very little or no oversight of the contractor by the customer.

2. All risks are borne by the contractor, who is paid upon delivery of a successful product.

Concurrent Engineering

Concurrent (or simultaneous) engineering is a technique that addresses the management of total life cycle time (7,24,26), focusing on the single most critical resource in a very competitive market: time to market (or time to deployment). This is accomplished primarily by shortening the life cycle through the realization of three engineering subgoals:

1. Introduction of customer evaluation and engineering design feedback during product development.

2. A greatly increased rate of focused, detailed technical interchange among organizational elements.

3. Development of the product and creation of an appropriate production process in parallel rather than in sequence.

Concurrent engineering is a meta-process in which domain experts from all the departments concerned with developing a product at any stage of the life cycle work together as a Concurrent Engineering (CE) team, integrating all development activities into one organizational unit. The formation of the team does not, *per se*, shorten the engineering life cycle; however, through early involvement with the CE team, organizational learning and analysis activities

can be removed from the critical path to market. There is an explicit tradeoff of manpower for time to market. That is, the CE team involves more personnel for a greater fraction of the life cycle than in the case of the waterfall model. Although the CE team remains together for a greater percentage of the total life cycle, the life cycle is significantly shorter than in traditional models. Consuming a greater portion of a smaller resource may not increase cost and, in some cases, may actually decrease cost. However, the time to market may be greatly reduced. In terms of the abstract life cycle model, the activities labeled "recognize," "analyze," and "synthesize" can occur concurrently for all organizational elements involved in the development of the product. Of course, there will be some activities that have temporal, as well as logical, sequential dependence upon other activities (see Figure 2a and Figure 2b). Marketing, drawing upon organizational expertise, including RDT&E products, begins the process through the generation of an idea of a product, based upon market analysis. Marketing will generate targets for the selling price and the production costs of the proposed product to support management in deciding whether to proceed with product development. During development, the CE team work simultaneously with the design team, to generate in parallel a design for the manufacturing process.

A CE life cycle model is shown in Figure 3. The principal feature of this process model is the concurrent development of the product, the manufacturing process, and the manufacturing system through the continuous participation of the CE team (28). A notable feature of the life cycle is the absence of a return path from production to design. This deliberate omission is in recognition of the extremely high risk of losing market share because of engineering delays due to design errors.

The organizational response to a change to concurrent engineering from traditional methods is likely to be fraught with difficulty. An organization that has formed around a particular life cycle model, and that has experienced a measure of success, perhaps over a period of many years, will almost certainly resist change. Effort in several specific areas appears to be basic to any transition:

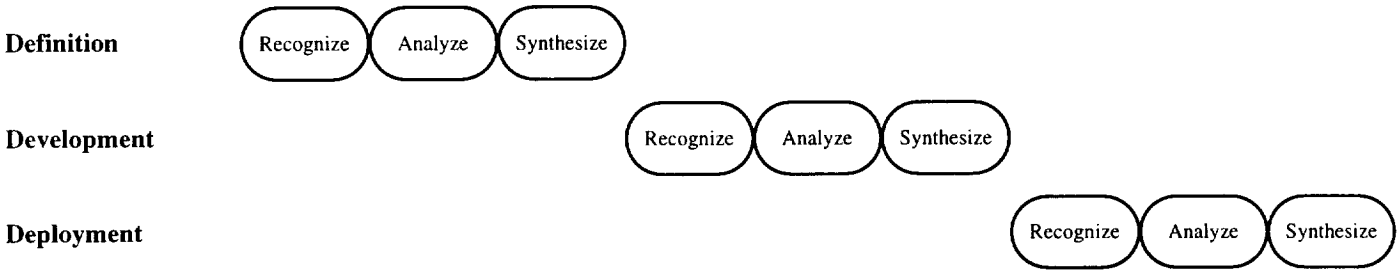


Figure 2a. Waterfall representation of abstract life cycle.

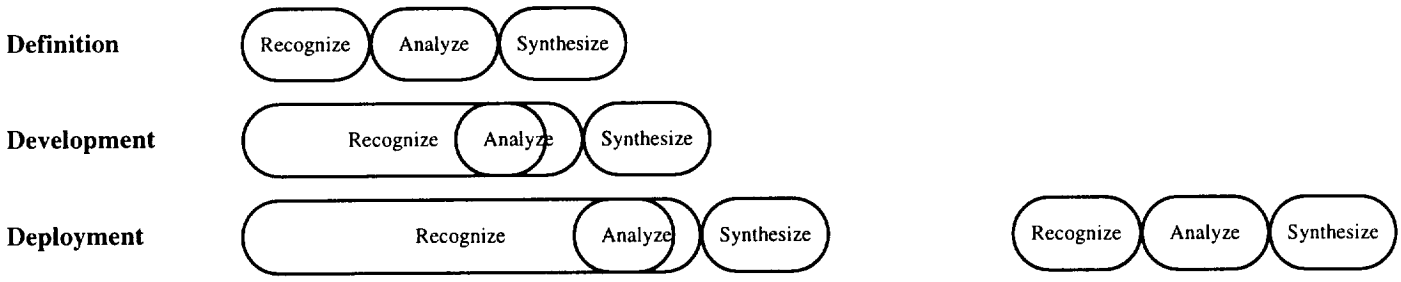


Figure 2b. The compressing effect of concurrent engineering upon the waterfall model.

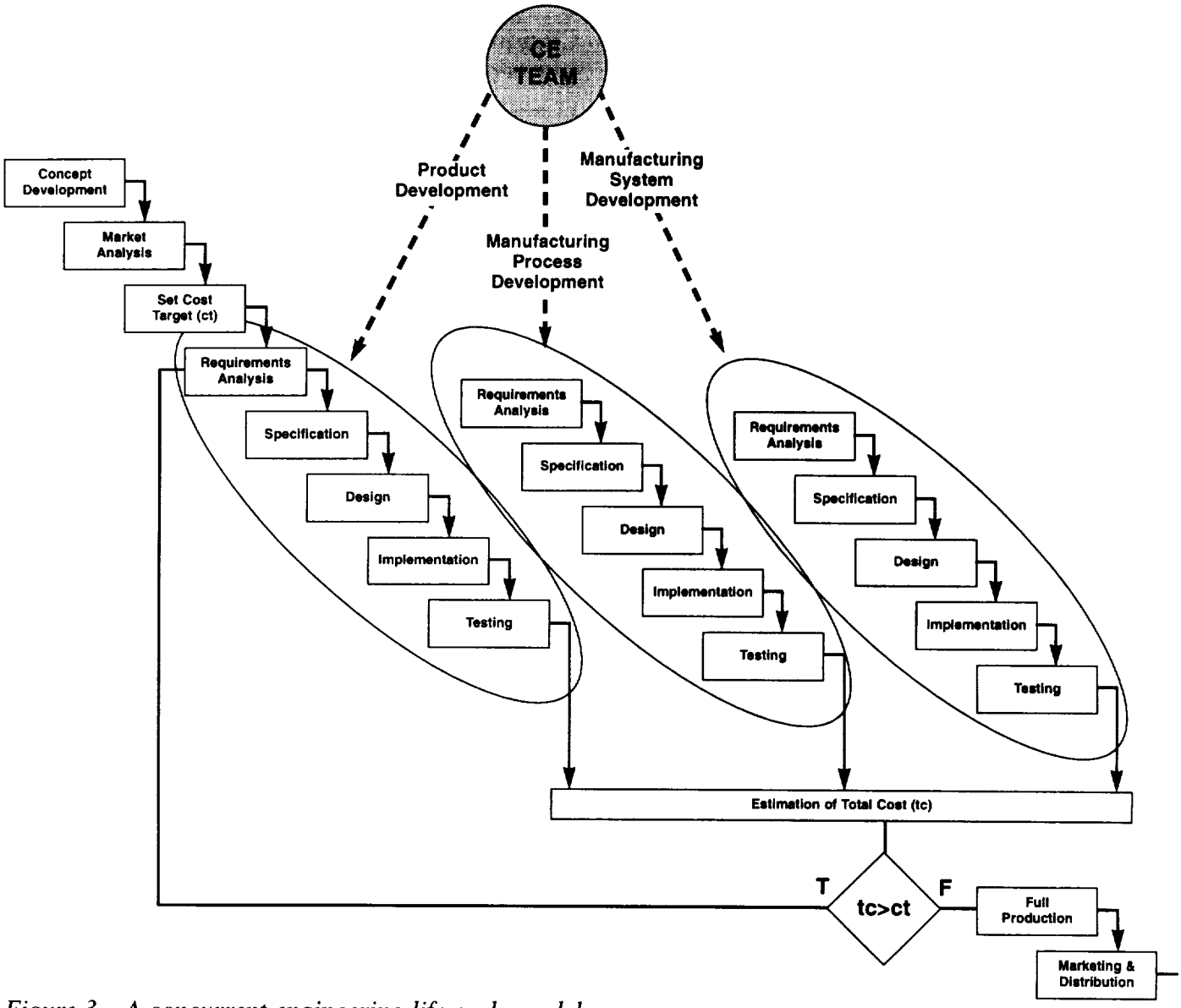


Figure 3. A concurrent engineering life cycle model.

- We have noted that life cycle models are purposeful, in that they reflect, organize and set into motion the organizational mission. A change of life cycle model should be accompanied by a clearly stated change of mission that may be digested, assimilated and rearticulated by all organizational elements — individuals, formal and informal team structures, and social groups.

- Formal team structures should be examined, destroyed and rebuilt, replaced or supplemented as necessary to conform to the new life cycle model. Informal team structures and individual expectation, such as those described by Stogdill, may be replicated and thus preserved by the formal organizational structure to minimize loss.

- Particular attention should be paid to intramural communication and cooperation among individuals, teams, and departments in the organization (7). Communication and cooperation are essential elements of concurrency. Acquisition or improved availability of communications tools, developed through

advances in communications technology, may reduce the cost and increase the rate of concurrency.

- Software reengineering for reuse.

Based upon two types of reuse identified by Barnes and Bollinger (2), it is useful to distinguish between two different types of software reengineering for reuse:

1. Software reengineering for maintenance (adaptive reusability) improves attributes of existing software systems, sometimes referred to as legacy systems, that have been correlated to improvements that improve software maintainability (3).

2. Software reengineering for reuse (compositional reuse) salvages selected portions of legacy systems for rehabilitation to enable off-the-shelf reuse in assembling new applications (1).

Both types of reengineering for reuse share a common life cycle (20), shown in Figure 4, for reengineering to an object-oriented software architecture.

The life cycle is divided into three successive phases: reengineering concept development, reengineering product development, and deployment. During the concept development phase, reengineering is considered as an alternative to new software development. Considerations of scope and level of reengineering allow planning and cost estimation prior to the development phase.

Reengineering product development proceeds according to the scope and level of reengineering planned in the previous phase. Reverse engineering of the old software is followed by forward engineering to create a new product. During the reverse engineering stage, products are recreated at the design and specification levels as needed to recapture implementation decisions that may have been lost over the lifetime of the legacy software. During the entire reverse engineering stage, candidate objects are repeatedly created, modified or deleted as necessary to provide the basis of an object-oriented design for the forward engineering stage.

During the forward engineering stage, the candidate objects from the reverse engineering stage are used to create an object-oriented specification and design. Implementation through coding and unit testing complete the development phase. During the deployment phase, software integration and testing, followed by system integration and testing, allow production and deployment to proceed.

Software reengineering is often associated with business process reengineering. A recent study (21) shows that there is a reciprocal relationship between business process reengineering and software reengineering. The enabling role of information technology makes possible the expansion of the activities of business processes. Moreover, changes in support software may influence changes in the business process. In particular, changes in support software make the software more useful or less useful to a given business process, so that the business process

Reengineering Concept Development			Reengineering Product Development						Deployment		
			Reverse Engineering			Forward Engineering					
Feasibility	Scope	Level of Reengineering	From Code to Design	From Design to Specs	Identification of Candidate Objects	Respecification	Design	Coding and Unit Testing	Software Integration and Testing	System Integration and Testing	Production and Deployment

Figure 4. Software Reengineering Life Cycle.

(or, indeed, the software) must adapt. Changes in the potential for software functionality that are due to improvements in the technology may enable changes in business processes, but must not drive them. In general, successful technology follows, rather than leads, humans and organizations.

Similarly, changes in the business process create changes in the requirements for support software. However, this cause-and-effect relationship between business process reengineering and software reengineering cannot be generalized and is inherently unpredictable. Each case requires independent analysis. The effect of business process reengineering on software may range from common perfective maintenance to reconstructive software reengineering. New software may be required in the event that the domain has changed substantially. Because the software exists to automate business process functions, the purpose of the support software may be identified with the functions comprised by the process. Therefore, reengineering the process at the function level will in general always require reengineering the software at the purpose level. This is equivalent to changing the software requirements. Software reengineering can be the result of business process reengineering, or it can be the result of a need to improve the cost-to-benefit characteristics of the software. An important example of software reengineering that may have little impact on the business process is the case of reengineering function-oriented software products into object-oriented products, thereby choosing the more reactive paradigm to reduce excessive cost due to poor maintainability.

There are many levels of business process reengineering and of software reengineering, ranging from re-documentation to using business process reengineering as a form of maintenance. In both there is a continuum between routine maintenance (minor engineering) and radical, revolutionary reengineering. At both ends of the spectrum, change should be engineered in a proactive, not a reactive manner. As Sage notes, reengineering ". . . must be top-down directed if it is to achieve the significant and long-lasting effects that are possible. Thus, there should be a strong, purposeful and systems management orientation to reengineering" (25).

Lowry and Duran (18), in assessing the adequacy of the waterfall model, cite the lack of adequate support for incremental development, such as many artificial intelligence applications. The spiral model is much more natural, especially as computer-aided software engineering (CASE) tools shorten the production cycle for the development of prototypes. In terms of the spiral model (4), the amount of time needed to complete one turn of the spiral has been shortened for many of the turns through the use of CASE tools. A potentially greater savings can be realized by reducing the number of turns in the spiral as well as through the development of knowledge-based tools. Lowry believes that much of the process of developing software will be mechanized through the application of artificial intelligence technology (17). Ultimately, the specification-to-design-to-code process will be replaced by domain-specific specification aids that will generate code directly from specifications. This interesting vision is based upon much current reality, not only in the CASE arena, but also in the area of domain-based reuse repositories (11,16,19). In terms of the life cycle implications, the waterfall will become shorter, and the spiral will have fewer turns.

(1) Robert S. Arnold and William B. Frakes, "Software Reuse and Re-engineering," in *Software Reengineering*, R. S. Arnold, Editor. Las Alamitos, CA: IEEE Computer Society Press, 1992, pp. 476-484.

(2) Bruce H. Barnes and Terry B. Bollinger, "Making Reuse Cost-Effective," *IEEE Software*, pp. 13-24, January, 1991.

(3) Victor R. Basili, "Viewing Maintenance as Reuse-Oriented Software Development," *IEEE Software*, pp. 19-25, January, 1990.

(4) Barry W. Boehm, *Software Engineering Economics. Prentice-Hall Advances in Computing Science and Technology Series*, R. T. Yeh, Editor. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

(5) Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*. Reading, Massachusetts: Addison-Wesley Publishing

Company, Inc., 1975 (reprinted with corrections, January 1982).

(6) Vernon E. Buck, "A Model for Viewing an Organization as a System of Constraints," in *Approaches to Organizational Design*, 1971 paperback edition, J. D. Thompson, Editor. Pittsburgh: University of Pittsburgh Press, 1966, pp. 103-172.

(7) Donald E. Carter and Barbara Stilwell Baker, *Concurrent Engineering: The Product Development Environment for the 1990s*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1992.

(8) P. B. Crosby, *Quality is Free*. New York: McGraw-Hill Book Company, 1979.

(9) Kingsley Davis, *Human Society*. New York: The Macmillan Company, 1949.

(10) Department of Defense, *Software Development and Documentation (MIL-STD-498)*, 5 December 1994 edition. Washington, DC: United States of America, 1994.

(11) John E. Gaffney Jr. and R. D. Cruickshank, "A General Economics Model of Software Reuse," in *14th Proceedings of the International Conference on Software Engineering*. New York: Association for Computing Machinery, 1992.

(12) Watts S. Humphrey, *A Discipline for Software Engineering*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1995.

(13) Watts S. Humphrey, *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989 (reprinted with corrections August, 1990).

(14) Darrel Ince, *ISO 9001 and Software Quality Assurance*. Maidenhead, Berkshire, England: McGraw-Hill Book Company Europe, 1994.

(15) International Benchmarking Clearinghouse, *Assessing Quality Maturity: Applying Baldrige, Deming, and ISO 9000 Criteria for Internal Assessment*. Houston, Texas: American Productivity and Quality Center, 1992 (American Productivity and Quality Center, 123 North Post Oak Lane, Houston, Texas 77024).

(16) Wojtek Kozaczynski, "The 'Catch 22' of Reengineering," in *12th International Conference on Software Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1990, catalog number 90CH2815-2 (IEEE), p. 119 (26-30 March).

(17) Michael R. Lowry, "Software Engineering in the Twenty-First Century," *AI Magazine*, volume 14, number 3, pp. 71-87, Fall, 1992 (Also appeared in Automating Software Design, eds. M. Lowry and R. McCartney, MIT/AAAI Press, 1991).

(18) Michael R. Lowry and Raul Duran, "A Tutorial on Knowledge-Based Software Engineering," in *The Handbook of Artificial Intelligence*, volume IV, A. Barr, P. R. Cohen, and E. A. Feigenbaum, Editors. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1989, chapter XX.

(19) John M. Moore and Sidney C. Bailin, "Domain Analysis: Framework for Reuse," in *Domain Analysis and Software System Modeling*, R. Prieto-Diaz and G. Arango, Editors. Los Alamitos, California, USA: IEEE Computer Society Press, 1991.

(20) Floyd Guyton Patterson Jr., "Reengineerability: Metrics for Software Reengineering for Reuse," Ph.D. Dissertation, George Mason University (Fairfax, Virginia, USA), 1995.

(21) Floyd Guyton Patterson Jr., "A Study of the Relationship Between Business Process Reengineering and Software Reengineering," *Information and Systems Engineering*, volume 1, number 1, pp. 3-22, March, 1995.

(22) Daniel J. Paulish and Anita D. Carleton, "Case Studies of Software-Process-Improvement Measurement," *IEEE Computer*, volume 27, pp. 50-57, September, 1994.

(23) John T. Rabbitt and Peter A. Bergh, *The ISO 9000 Book: A Global Competitor's Guide to Compliance and Certification*, 2nd edition. New

York: AMACOM Division of American Management Association, 1994.

(24) Milton D. Rosenau Jr., *Faster New Product Development: Getting the Right Product to Market Quickly*. New York: AMACOM Division of American Management Association, 1990.

(25) Andrew P. Sage, *Systems Management for Information Technology and Software Engineering*. New York: John Wiley and Sons, 1995.

(26) Bernard N. Slade, *Compressing the Product Development Cycle*. New York: AMACOM Division of American Management Association, 1993.

(27) Software Engineering Institute, *Capability Maturity Model* (CMU/SE-91-TR-24). Pittsburgh, PA: Carnegie Mellon University, 1991.

(28) Raymond T. Yeh, "Notes on Concurrent Engineering," *IEEE Transactions on Knowledge and Data Engineering*, volume 4, number 5, pp. 407-414, 1992.

Excerpted and modified from "Systems Engineering Life Cycles," by F. G. Patterson Jr., in *Handbook of Systems Engineering*, Chapter 1, A. P. Sage and W. Rouse, eds., New York: John Wiley & Sons, Inc., 1996 (to appear). Used with permission of the publisher.

Software Reuse in Wind Tunnel Control Systems

by Charles E. Niles

Software is an important element of wind tunnel operations at NASA Langley Research Center (LaRC). Reuse of wind tunnel automation software, while limited, has produced benefits on a local scale. Two forms of reuse have been utilized—from project to project, and from system to system within a project.

LaRC possesses a broad range of wind tunnels, test facilities and laboratories which support our nation's aeronautics and space endeavors. The wind tunnels enable researchers to study aerodynamics, fluid dynamics, acoustics, heat transfer and other similar interests in order to evaluate and improve the performance of aircraft, missiles, jet engines, spacecraft and various components thereof.

Although wind tunnels vary in purpose, size, shape and operating range, similar software has been used across facilities for data collection systems, data reduction systems and control systems.

To simplify discussions, only closed-circuit wind tunnels will be addressed. Essentially, a wind tunnel is a continuous, large-diameter cylinder arranged in an elongated, oval circuit. Air or some other medium is moved at speeds up to Mach 1.3 around the circuit by large fan blades. Two-thirds of the distance around the circuit from the fan blades is a test section in which a model is manipulated at various pitch and roll angles to gather data on aerodynamic effects. The model may be fitted with scale-size jet engines which are driven through independent high pressure air systems. Pressures inside the tunnel range from one to several atmospheres. Temperatures range from near absolute zero to over 150 degrees Fahrenheit.

Wind tunnel control systems actuate subsystems to affect fan speed, pressure, temperature, pitch, roll and other tunnel, model or test article processes. The dynamic interaction among tunnel processes causes control system software to be moderately complex and uniquely adapted and tuned for a facility. Personnel safety and facility/model protection justifies very reliable software which further adds to complexity.

Automation System Projects

Between early 1985 and late 1989, the 16-Foot Transonic received a major overhaul in which almost every operational system was affected. The staff assembled for this effort encompassed every imaginable engineering discipline—civil, electrical, structural, mechanical, controls, and software, to name a few.

During the overhaul, the control room was modernized and state-of-the-art microcontroller equipment was installed to provide automated controls for tunnel (TNL), model attitude (MDL), and high pressure air (HPA) systems which were interconnected with a standby (STB) system via a local ethernet network. The STB system served as a ready standby for any one of the other three microcontroller systems. The STB shared its chassis with a gateway (COM) which communicated with a process monitor and control (PMC) minicomputer via a custom parallel link. The PMC also communicated with the facility data acquisition system via a separate network.

The effort involved the development of over 150,000 lines of code. The TNL, MDL, HPA, and STB microcontroller code was written in PLM, FORTRAN, and assembly language for Intel 8086 CPUs running the RMX-86 operating system on the Multibus I architecture. The PMC code was written entirely in FORTRAN-77 for a Modcomp running the MAX IV operating system.

Approximately 75% of the PLM source code, known as the environment code, is identical on the TNL, MDL, and HPA systems. The FORTRAN code,

which represents the control algorithms, is necessarily unique. Overall, PLM (85%) and FORTRAN (15%) make up the bulk of the code. Assembly language (less than 1%) was used only when necessary. The STB system is a composite of the environment code and the specific portions of the TNL, MDL, and HPA code.

The TNL, MDL, HPA, and STB systems consist of about 30,000 source lines each, while the COM and PMC combine for about 30,000 lines. During the project, the environment code was developed using the TNL system as the basis. In effect, the MDL, HPA, and STB systems were instances of reuse within the project. In addition, that portion of the COM software which supports the local network among the systems is identical. The COM software which supports the parallel link and the PMC have not been reused.

A year after the 16-Foot project, two control system upgrade projects were performed. The first project, at the Jet Exit Test Facility, involved cloning the HPA system, making some facility-specific changes, and enhancing the environment code. This was a small project, requiring one software developer. In an extremely unusual scenario, the computer hardware arrived, the control system software was installed and checked out staticly. Several months later, the rest of the project caught up while the software developer was at graduate school. The author, a veteran of the 16-Foot project, was pressed into service to support checkout of the integrated system. As a testament to the stability of the software, checkout went smoothly.

The second project at the National Transonic Facility (NTF) was more complex. This project included newer hardware (80486 Multibus II vs. 8086 Multibus I), a newer operating system (RMX-III vs RMX-86), a newer language (PLM-386 vs PLM), and integration of the existing control algorithms. Each element offered a different challenge. The first three were straightforward—new hardware required new drivers, a new operating system required new or modified system calls, and a new language was almost transparent. But the existing control algorithms had to be repackaged to conform to the environment-algorithm interface.

In the transition from 16-Foot to NTF, some parts of the environment code were optimized. However, adding generalized code which was formerly handled uniquely by the microcontrollers caused the overall size of the environment code to increase slightly. At the end of the NTF project, 90% of the original 16-Foot microcontroller software had been reused with little or no modification.

In all, these three projects account for eight systems which consist of the same environment code with incremental improvements and optimization over time.

Issues

The obvious benefits of reusing the environment code include shorter product delivery time, minimal time invested in documentation after the initial facility, and easier maintenance. Beyond the obvious benefits, the environment code has provided a firm foundation to which new control algorithms have been and are being added.

It is a pity that the code has not been reused more. Unfortunately, the large initial investment in the 16-Foot project took its toll. When the project was completed two years late, management took a dim view of performing a software intensive project using in-house personnel. Thus, only selected projects were subsequently tackled. Of course, there were other reasons—more projects than in-house staff could perform, urgency in obligating funding (a form of sheer madness), the emergence of commercial applications, and a rapidly changing hardware climate.

There have been several significant impediments to reuse. Perhaps the biggest has been individuality. Most of NASA's major accomplishments are attributable to individuals. NASA is full of free-thinking scientists and engineers...and software developers who are probably the most individualistic of all. Of course, management traditionally has fostered an environment of creativity and designer-preference.

So, there has been little discipline to reuse anything, much less software, except on an individual basis.

Another impediment is organizational structure. LaRC lags far behind the NASA space centers where software has been a critical element of most everything ever launched. Although there are pockets of individual software expertise scattered across LaRC, there is no formal organization. Software reuse will flourish more in a software engineering organization than not. In the author's engineering organization, mechanical, structural, and civil disciplines are prevalent. There are engineers who develop software, but no software engineers. Such an environment is simply not conducive to software engineering. Without software engineering, good software design occurs by accident. Usually, inferior design results in inferior source code which should not be reused.

Next Generation

Three events since mid-1994 have changed the long-term vision of automation projects within the author's facility automation software development staff. First, the staff underwent a capability self-assessment which was facilitated by a cross-center team of experienced software personnel and the Navy's software engineering group at Damn Neck, Virginia. The results brought management attention to issues which affected the broader organization. In short, the staff should focus more on facility automation and less on software development.

Second, the author, long an advocate of a standard approach to automation systems at LaRC wind tunnels, accepted an offer for a team from the IV&V Center in West Virginia to conduct a domain engineering effort of wind tunnel control systems. The team, known as the Software Optimization and Reuse Team (SORT), methodically analyzed several wind tunnel control systems and developed an essential set of requirements. More recently, SORT has developed a set of derived requirements during a domain design. The SORT effort also influenced the third event.

Third, the author and a systems engineer who is also a standard product advocate have embraced a new approach with the full support of management who want to reduce development costs and time in the face of budget cuts and loss of personnel.

The rapidly changing hardware climate has also been a factor. Since Intel decided to drop support for its Multibus II architecture (along with the RMX-III operating system), the need for a different hardware platform became evident. The VXI bus architecture has been chosen in order to accommodate the instrumentation needs of both control and data acquisition systems. The Lynx Operating System (LynxOS) has been chosen to replace RMX-III. Together, they represent the standard hardware/operating system platform of the next generation.

The next generation also involves a widely used software package known as EPICS (Experimental Physics and Industrial Control System). EPICS, which originated within the Department of Energy, was developed by computer scientists and physicists for application to electron beam accelerator facilities. Over the years, EPICS has been adapted to other applications including physics labs, astronomy labs, and jet engine test facilities. Although EPICS is making its first appearance at LaRC, it is already installed at over 75 sites worldwide. Oddly enough, EPICS is used by NASA at the Canberra tracking station.

The author's organization believes EPICS will transcend software reuse. Combined with a standard hardware platform, EPICS means that **systems** can be reused.

Resources for NASA Managers

by W. M. Lawbaugh

Healing the Wounds
by David M. Noer
San Francisco: Jossey-Bass, 1993

"Overcoming the Trauma of Layoffs and Revitalizing Downsized Organizations" is the subtitle of this well-designed book. The author's purpose is twofold: "to explain the nature of layoff survivor sickness and to help both individuals and organizations formulate strategies to fight off this disease."

"Layoff survivor sickness" is Noer's term for the widespread and toxic fear, anger and depression that follow massive layoffs from downsizing, restructuring, mergers and reengineering. He documents the "survivor syndrome" from research on atomic bomb survivors, Nazi concentration camp survivors and even survivors in the space shuttle program after the Challenger disaster. He finds "guilt, anxiety and fear" as symptoms.

Healing the Wounds follows roughly the five stages of grieving made popular by Dr. Elizabeth Kubler-Ross: denial to anger to bargaining to guilt and depression and finally to letting go and acceptance. The pivotal chapter is nine when Noer urges: "break the codependency chain and empower people." This new-paradigm behavior was characterized a year before Noer published this book, when Dagwood Bumstead (the prototype of the old employment contract employee) tells Mr. Dithers that he is quitting the corporation to take a job in Blondie's entrepreneurial catering business. (Two weeks later, however, the comic strip shows Blondie firing Dagwood for eating her profits and Dagwood returning sheepishly to the J.C. Dithers Co.)

Nevertheless, Chapter 9 shows the folly of those who measure their self-worth by their success in the codependent organizational system. "Don't place your spiritual currency in the organizational vault," Noer pleads. Instead, he advises, let go of the codependent relationship with the abusive organization, seek detachment through good work (what Paul Hirsch calls "free agent management" in *Pack Your Own Parachute*, 1987) and try to "connect with a core purpose," such as the spiritual awakening that comes with completion of a Twelve-Step Program.

David Noer, vice president for training and education the Center for Creative Leadership, says that life after downsizing can be revitalizing for both the individual and the organization if and when codependency yields to autonomy and self-empowerment. If not, the organization is in decline.

To Build the Life You Want,
Create the Work You Love
by Marsha Sinetar
New York: St. Martin's Press, 1995

Entrepreneur Marsha Sinetar attempts to bring such corporate terms as reinventing and reengineering down to a personal level. Her purpose is to provoke readers into finding their own "right livelihood," work done in service to humanity with pure intentions, according to Zen teaching. Or, in Jungian terms, to discover their individual "vocation" or calling to a higher level of life's work.

Alternating frequently between Eastern and Western holistic definitions of work, Sinetar claims that "our new job security requires healthy entrepreneurial prowess." In other words, job security depends less upon the employer and more upon the worker's own self-reliance, creative resources and enthusiastic engagement (what Rollo May calls "creative encounter") with meaningful work.

To Build the Life You Want is based upon Abraham Maslow's hierarchy of values, the apex of which is "self-actualization," or, in her terms, "vocational integration." Her ample quotations and anecdotes come from a wide variety of sources, from Zen

Masters and Whoopi Goldberg to Edward deBono and Louis Lahr, former CEO of 3M. Most of all, she quotes herself, especially an earlier book, *Do What You Love, The Money Will Follow.*

Sinetar, who left a secure job as public school administrator to strike out on her own in business and writing, does have some interesting perspectives. For example, in her final chapter she invites the reader to view "work as art," with all its attendant risks, trials, discipline and creativity.

"The most productive entrepreneurs I know *gain* energy by managing it," she says in another chapter. Instead of heavy lunches she recommends physical workouts, yoga, massage, meditation, breathing exercises or fruit juices for better health and vitality. "Vitality also translates into widened opportunities: People like to be around us when we're centered and enthused."

Much of this book is based upon common sense, but it is practical. Each chapter ends with "A Summary Strategy" with thought-provoking questions and exercises that could keep you busy for hours. Lively anecdotes, crisp interviews and bountiful suggestions for deeper reading add to the enjoyment of this little (200 pages) book.

Being Digital
by Nicholas Negroponte
New York: Vintage Books, 1996

Nicholas Negroponte is founding director of the Media Lab at M.I.T. and a frequent contributor to *Wired* magazine, where portions of this book appeared first.

The author distinguishes between being digital (all-at-onceness, connected, now) and being analog (one thing at a time, fragmented, left-brained). He is decidedly pro-technology and insists that "many electronic games teach kids strategies and demand planning skills that they will use later in life."

Although *Being Digital* is available on tape AudioBooks and on CD, both in abridged forms, it is somewhat ironic it is published in "atoms," on paper, not in cyberspace. The author admits he does not like to read, owing mainly to his dyslexia.

Nevertheless, he has produced one of the clearest, most interesting guides to the cyberworld he helped to create. He explains bits, bytes and bandwidth, data compression, high-definition TV, and some of the myths and half-truths surrounding each. Yet, while he discusses multimedia with competence, he admits his only use of the Internet is for electronic mail, not for research or data storage.

Perhaps the most interesting parts of *Being Digital* are Negroponte's fearless predictions of the digital future. He has already been proven correct in his assessment of foiled attempts to regulate and censor the Internet. Consider these prognostications:

- "I am convinced that by the year 2000 Americans will spend more time on the Internet . . . than watching television."
- CD-ROMs are "the Beta of the 90s," bound for extinction. "The fax machine . . . is a step backward."
- "The value of information about information can be greater than the value of information itself." Witness *TV Guide*. Information in the future will be customized and personalized to interface well with the consumer.
- "Digital life will include very little real-time broadcast" except for sports and elections.
- "In future media there will be more pay-per-view," not less, unless you prefer advertising.
- "The notion of an instruction manual is obsolete . . . nothing short of perverse."
- "The middle ground between work and play will be enlarged dramatically." Ditto for love and duty.
- "By the year 2020, the largest employer in the developed world will be 'self.' Is this good? You bet."

There is a dark side to being digital, and Negroponte glosses over privacy invasion, copyright piracy and radical worker dislocation. Nevertheless, anyone whose 80-year-old mother sends him email daily cannot be less than optimistic about the future.

The End of Work
by Jeremy Rifkin
New York: Putnam's, 1995

"The Decline of the Global Labor Force and the Dawn of the Post-Market Era" is the subtitle of Jeremy Rifkin's latest book. His earlier books, *The Emerging Order* and *Time Wars*, were well received and successful. This one is ominous, especially in its prediction of global unemployment.

"The Information Age has arrived," Rifkin announces. "In the years ahead, new, more sophisticated software technologies are going to bring civilization even closer to a near-workerless world." In the past, fading sectors of the economy always seemed to give way to new, emerging sectors. Agriculture gave way to manufacturing, which gives way to the service sector. However, Rifkin notes, all three sectors are experiencing "technological displacement."

If there is a new economic sector emerging, it is the "knowledge sector," made up of entrepreneurs, computer programmers, educators and consultants. This small but growing sector of "knowledge workers"cannot begin to absorb more than a fraction of the millions of workers displaced by machines.

Especially hard hit during re-engineering and downsizing were African Americans in office and clerical jobs and laborers, as well as trade unionists, like Machinists, Steelworkers and UAW members. Some entire unions, such as typographical workers, were wiped out in the recent postindustrial revolution.

Rifkin points out that this is not a national problem. At times his rhetoric is harsh, scary. For example:

> The death of the global labor force is being internalized by millions of workers who experience their own individual deaths, daily, at the hands of profit-driven employers and a disinterested government. They are the ones who are waiting for pink slips, being forced to work part-time at reduced pay, or being pushed onto the welfare rolls. With each new indignity, their confidence and self-esteem suffer another blow. They become expendable, then irrelevant and finally invisible in the new high-tech world of global commerce and trade.

The price of this "program," for Rifkin, is threefold: a wider disparity between the super-rich and the abject destitute, a slow death for the middle (working) class, and "a more dangerous world" due to teen unemployment and terrorism.

Like most of his hard-hitting, penetrating and well documented books on social policy, this book offers solutions to turn the gloom and doom into sunshine and happiness. His solutions are two. First, we must reengineer the work week as we have the workplace. Europeans are experimenting now with the shorter work week (four days) or 30 hours per week. Secondly, Rifkin proposes "a new social contract" based upon "empowering the third sector." Our market economies have stressed the government and commerce sectors at the expense of what he calls the social sector of education, health care, the arts, religion and community service. Various welfare reform schemes involve workforce in such third sector jobs as daycare and after-school programs.

Immediate implementation of both these solutions may, however, be too little too late, according to Rifkin. Some social scientists imagine a high-tech world just around the corner wherein two percent of the world's population can sustain the food, shelter and clothing needs of all the rest. What the unemployed do in such a world is still a mystery, but Rifkin produces plenty of evidence that the end of work as we know it may substantially diminish if not disappear yet within our own lifetime.